

ISTRUZIONI CONDIZIONALI E CICLI

[1]

○ ISTRUZIONI CONDIZIONALI

- ⇒ Esigenza di ramificare l'esecuzione di un programma
- ⇒ Esigenza di eseguire particolari parti di codice a seconda del verificarsi di determinate condizioni
- ⇒ Esigenza di effettuare test su particolari valori

○ CICLI

- ⇒ Ripetizione condizionata di porzioni di codice
- ⇒ Esecuzione di una sequenza ripetitiva di azioni predeterminata o secondo condizione
- ⇒ Es: Scorrimento di array, ordinamento, ricerca di un elemento.

ISTRUZIONI CONDIZIONALI

[1]

○ IF-ELSE [1]

- ⇒ Valuto un'espressione ed eseguo un'istruzione piuttosto che l'altra a seconda del suo valore di verità
- ⇒ Valuto se eseguire o meno alcune istruzioni
- ⇒ Formato:

```
if (espressione)
    istruzione_1
else
    istruzione_2
```
- ⇒ La parte con **else** e' opzionale e, se ve ne sono piu' di uno innestati, si riferisce all'ultimo if
- ⇒ Se si vogliono eseguire piu' istruzioni si devono racchiudere tra {...} (blocco di istruzioni)

ISTRUZIONI CONDIZIONALI

[2]

○ IF-ELSE [2]



Es 1:

```
[...]
if (a>b)
    printf("a è maggiore di b");
else
    printf("a è minore o uguale al numero b");
[...]
```



Es 2:

<pre>[...] if (n>0) if (a>b) z=a; else z=b; [...]</pre>	<pre>[...] if (n>0) { if (a>b) z=a; } else z=b; [...]</pre>
---	---

ISTRUZIONI CONDIZIONALI

[3]

○ ELSE-IF [1]



Raffino con ulteriori test una condizione di if



Formato:

```
if (espressione)
    istruzione_1
else if (espressione)
    istruzione_2
[...]
```

else

```
    istruzione_n
```



Puo' essere utilizzato per valutare tre o piu' scelte possibili ed eseguire operazioni di conseguenza.

ISTRUZIONI CONDIZIONALI

[4]

○ ELSE-IF [2]

↪ Es costruito else-if:

```
if (a>b)
    printf("a è maggiore di b");
else if (a<b)
    printf("a è minore di b");
else
    printf("a è uguale b");
```

↪ Es equivalente con if:

```
if (a>b)
    printf("a è maggiore di b");
else
{
    if (a<b)
        printf("a è minore di b");
    else
        printf("a è uguale b");
}
```

ISTRUZIONI CONDIZIONALI

[5]

○ SWITCH [1]

↪ Permette di eseguire diverse istruzioni a seconda del particolare valore di un'espressione

↪ Formato:

```
switch (espressione)
{
    case espress_cost1: istruzioni1; break;
    case espress_cost2: istruzioni2; break;
    [...]
    default : istruzioni;
}
```

↪ Utile avere l'opzione di default

↪ Ogni case termina con break altrimenti eseguirei anche i successivi

ISTRUZIONI CONDIZIONALI

[6]

○ SWITCH [2]

↪ Es 1:

```
[...]
switch(CarattereLetto)
{
    case 'A': case 'B': case 'C': printf(" A B o C ");
                                break;

    case 'D': printf(" D ");
              break;

    case 'E': printf(" ***** ");
              printf(" E ");
              break;

    default: printf(" Ne' A ne' B ne' C ne' D ne' E");
};
[...]
```

CICLI

[1]

○ WHILE [1]

↪ Ripete un blocco di istruzioni fino a quando è verificata la condizione di ingresso

↪ Ciclo a condizione iniziale

↪ Formato:

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

↪ Formato per una sola istruzione:

```
while (espressione)
    istruzione
```

○ WHILE [2]



Es 1:

```
#include <stdio.h>
main()
{
    float fahr, celsius;
    float lower, upper, step;
    lower=0.;
    upper=300.;
    step=20.;
    fahr=lower;
    while (fahr<=upper)
    {
        celsius=(5.0/9.0)*(fahr-32.0);
        printf("%3.0f %6.1f\n", fahr,celsius);
        fahr=fahr+step;
    }
}
```

○ FOR [1]



Ciclo di istruzioni con contatore



Formato:

```
for (esp1, esp2, esp3)
    istruzione;
```



Es: inizializzazione array di interi

```
for (i=0,i<MaxLungArray,i++)
    MioArray[i]=0;
```



Analogo ad un ciclo while dove:

```
esp1;
while (esp2)
{
    istruzione;
    esp3;
}
```

CICLI

[4]

○ FOR [2]



Es 1:

```
#include <stdio.h>
main()
{
    int fahr;
    float celsius;
    for(fahr=0; fahr<=300; fahr=fahr+20)
    {
        celsius=(5.0/9.0)*(fahr-32.0);
        printf("%3d %6.1f\n", fahr, celsius);
    }
}
```



Es 2:

```
[...]
for(fahr=0; fahr<=300; fahr=fahr+20)
    printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32.0));
[...]
```

CICLI

[5]

○ DO-WHILE [1]



Eseguo un gruppo di istruzioni almeno una volta e fino a quando e' verificata una condizione



Ciclo a condizione finale



Formato:

```
do
    istruzione;
while (espressione);
```



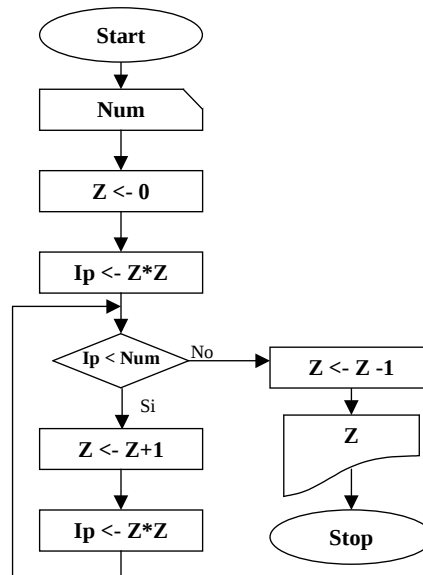
Equivalente al while a patto di eseguire le azioni una volta prima del ciclo (per tale motivo si usa poco)

ESEMPI SUI CICLI

[1]

○ RADICE INTERA [1]

↳ Diagramma di Jackson:



ESEMPI SUI CICLI

[2]

○ RADICE INTERA [2]

↳ Codice

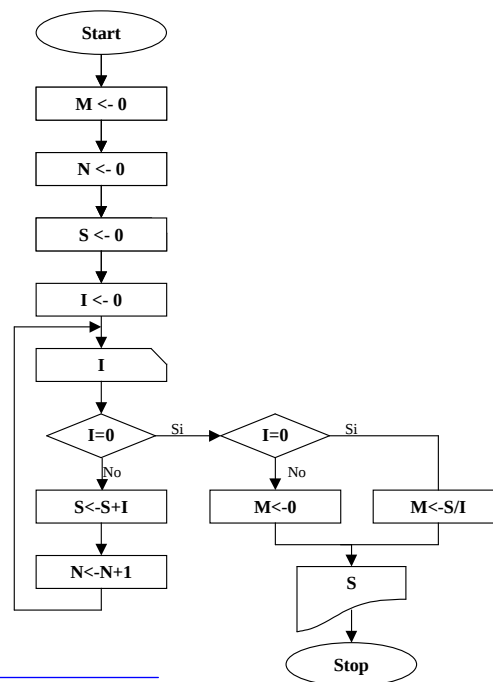
```
#include <stdio.h>
main()
{
    int Num,Z,Ip;
    printf("inserisci un numero: ");
    scanf("%d",&Num);
    Z=0;
    Ip=Z*Z;
    while (Ip<Num)
    {
        Z=Z+1;
        Ip=Z*Z;
    }
    Z=Z-1;
    printf("La radice intera di %d e' %d",Num,Z);
}
```

ESEMPI SUI CICLI

[3]

○ LETTURA SERIE DI NUMERI E CALCOLO MEDIA [1]

↳ Diagramma di Jackson



ESEMPI SUI CICLI

[4]

○ LETTURA SERIE DI NUMERI E CALCOLO MEDIA [2]

↳ Codice

```
#include <stdio.h>;
main()
{
    int N=0,S=0,I=0;
    float M=0.;
    printf("\nInserisci un numero: ");
    scanf("%d",&I);
    while(I!=0)
    {
        S=S+I;
        N++;
        printf("\nInserisci un numero: ");
        scanf("%d",&I);
    };
    if (N==0)
        M=0.;
    else
        M=(float)S/(float)I;
    printf("La media e' %f",M);
}
```

○ UN ALGORITMO DI ORDINAMENTO: BUBBLE SORT



Codice

```
#include <stdio.h>;
#define MaxNumEl 20;
main()
{
    typedef int Vettore[MaxNumEl];
    Vettore UnVettoreDiInteri;
    int disordinato=1, indice, temp;
    [...]
    while (disordinato)
    {
        disordinato=0;
        for(i=0,i<MaxNumEl-1;i++)
            if(UnVettoreDiInteri[i]> UnVettoreDiInteri[i+1])
            {
                temp= UnVettoreDiInteri[i];
                UnVettoreDiInteri[i]= UnVettoreDiInteri[i+1];
                UnVettoreDiInteri[i+1]= temp;
                disordinato=1;
            }
    }
}
```