

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Programmazione in C

Unità Cicli ed iterazioni

Cicli ed iterazioni

- La ripetizione
- Istruzione `while`
- Schemi ricorrenti nei cicli
- Istruzione `for`
- Approfondimenti
- Esercizi proposti
- Sommario

Riferimenti al materiale

➤ Testi

- Kernighan & Ritchie: capitolo 3
- Cabodi, Quer, Sonza Reorda: capitolo 4
- Dietel & Dietel: capitolo 4

➤ Dispense

- Scheda: "Cicli ed iterazioni in C"

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

La ripetizione

La ripetizione

- Concetto di ciclo
- Struttura di un ciclo
- Numero di iterazioni note
- Numero di iterazioni ignote

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```

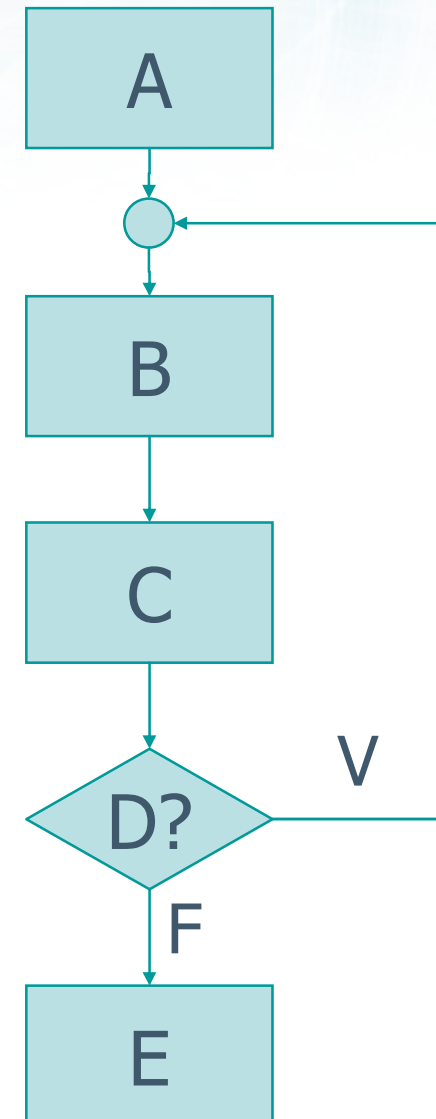


La ripetizione

Concetto di ciclo

Flusso di esecuzione ciclico

- È spesso utile poter **ripetere** alcune parti del programma più volte
- Nel diagramma di flusso, corrisponde a “tornare indietro” ad un blocco precedente
- Solitamente la ripetizione è controllata da una condizione booleana



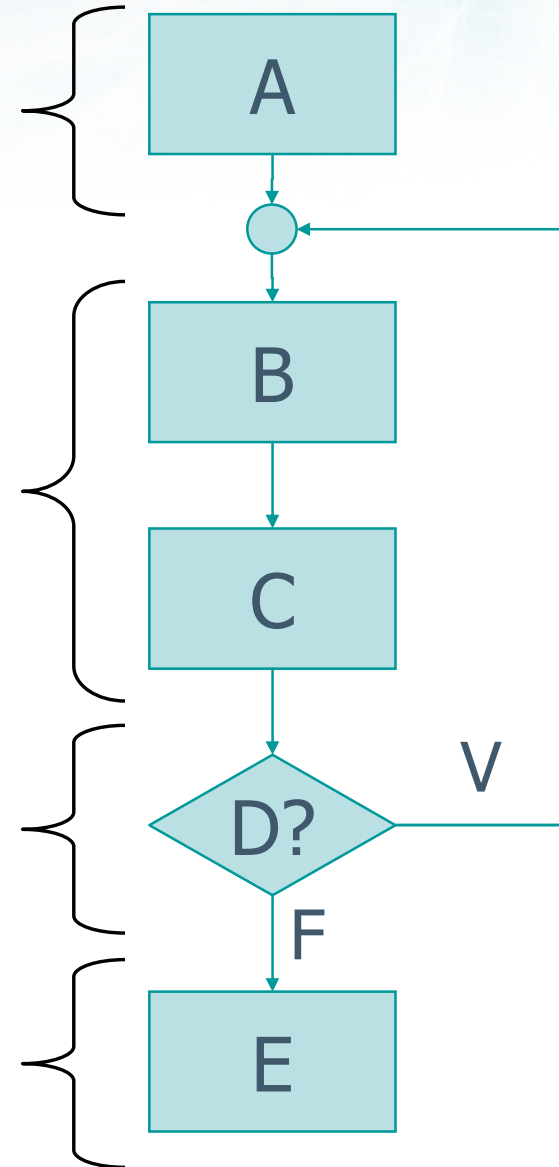
Flusso di esecuzione ciclico

Prima del
ciclo

Istruzioni
che vengono
ripetute

Condizione
di ripetizione

Dopo il ciclo





Errore frequente

- Ogni ciclo porta in sé il rischio di un grave errore di programmazione: il fatto che il ciclo venga ripetuto indefinitamente, senza mai uscire
- Il programmatore deve garantire che ogni ciclo, dopo un certo numero di iterazioni, venga terminato
 - La condizione booleana di controllo dell'iterazione **deve** divenire falsa

Istruzioni eseguibili ed eseguite

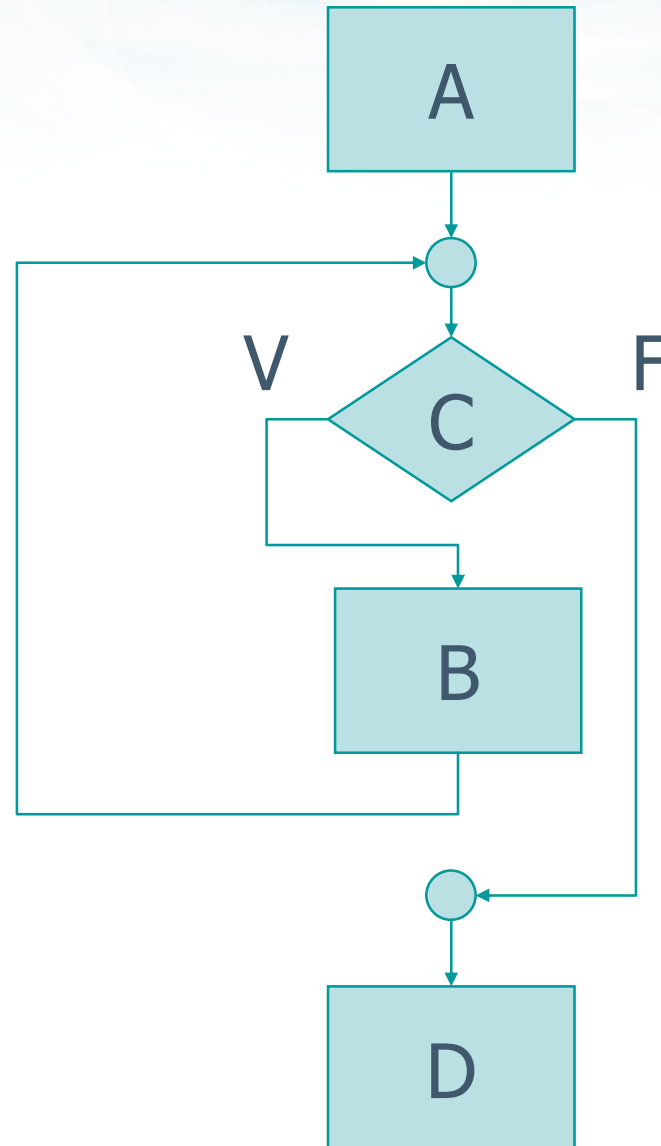
➤ Istruzioni eseguibili

- Le istruzioni che fanno parte del programma
- Corrispondono alle istruzioni del sorgente C

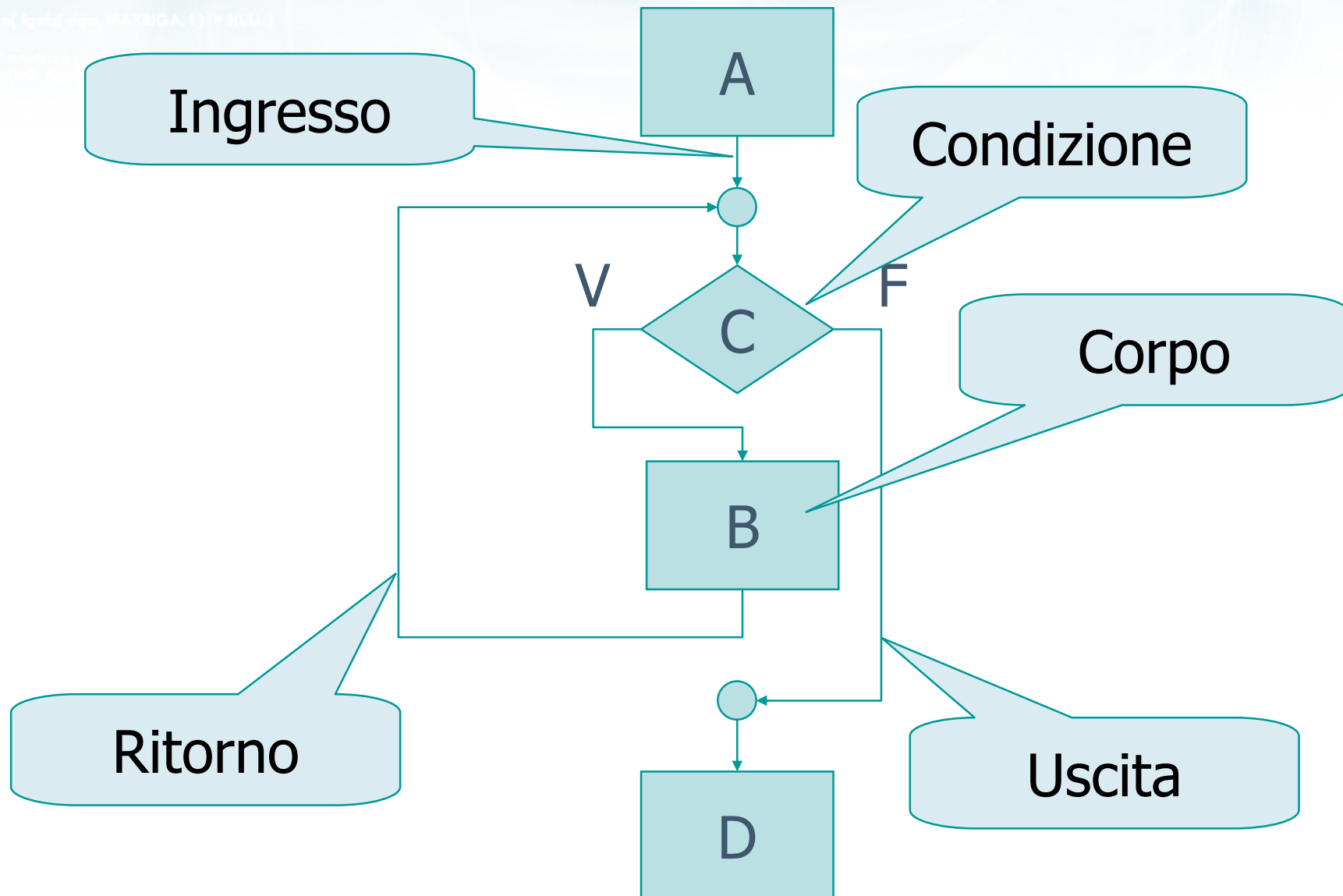
➤ Istruzioni eseguite

- Le istruzioni effettivamente eseguite durante una specifica esecuzione del programma
 - Dipendono dai dati inseriti
- Nel caso di scelte, alcune istruzioni eseguibili non verranno eseguite
- Nel caso di cicli, alcune istruzioni eseguibili verranno eseguite varie volte

Notazione grafica (while)

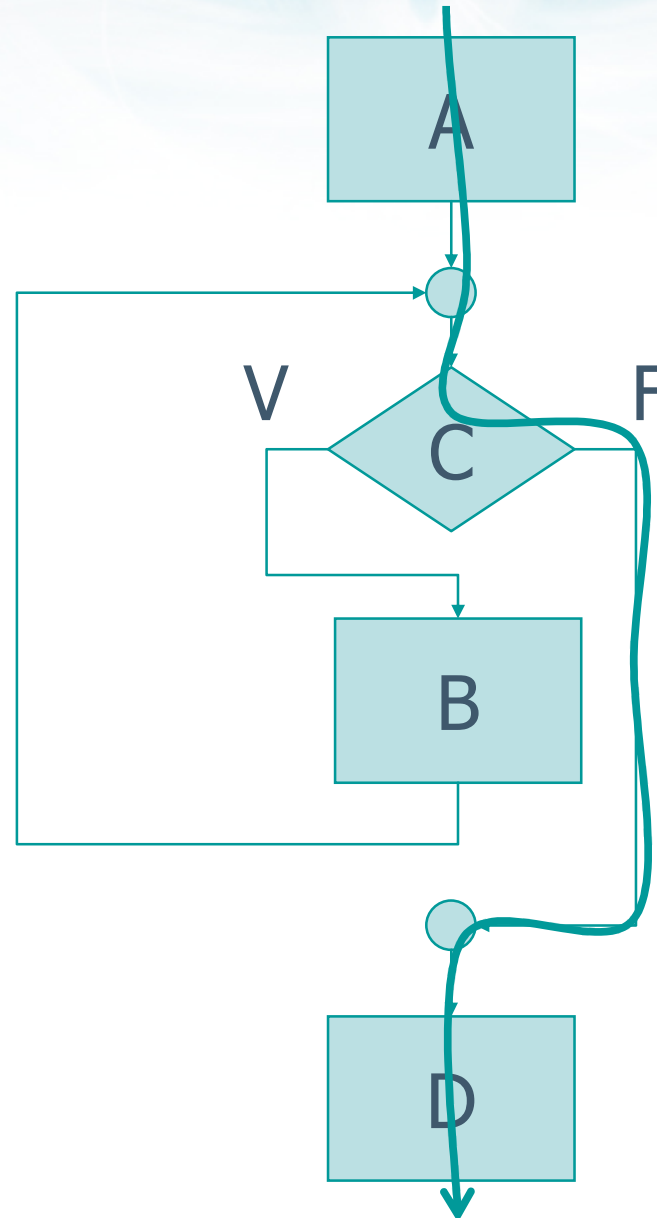


Notazione grafica (while)



Flussi di esecuzione

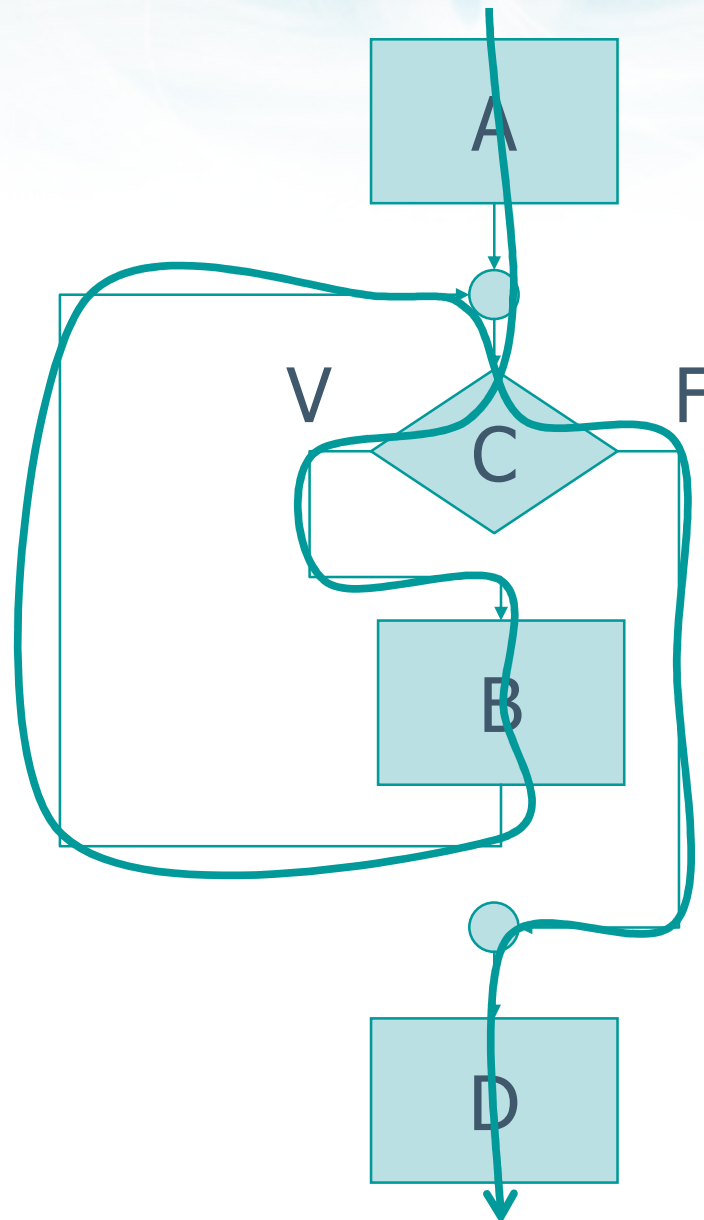
Zero iterazioni



A
C → Falso
D

Flussi di esecuzione

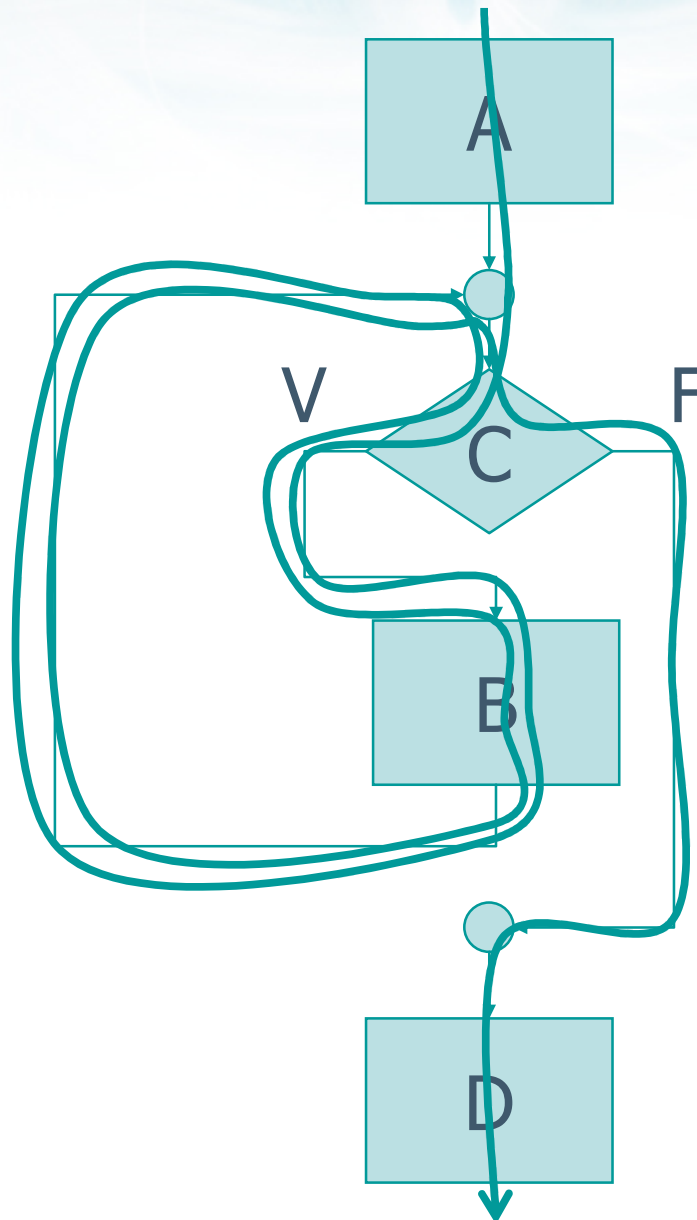
Una iterazione



A
C → Vero
B
C → Falso
D

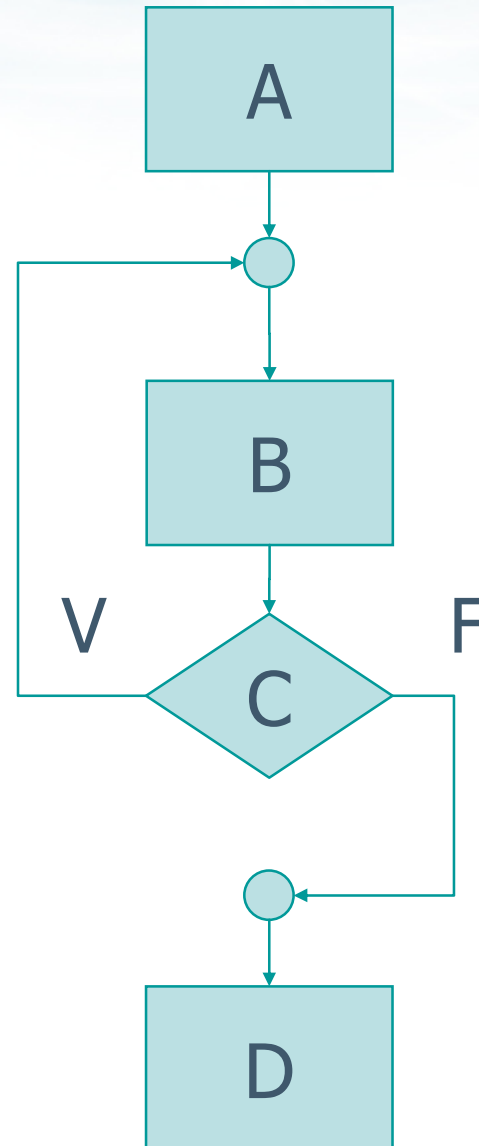
Flussi di esecuzione

Due iterazioni

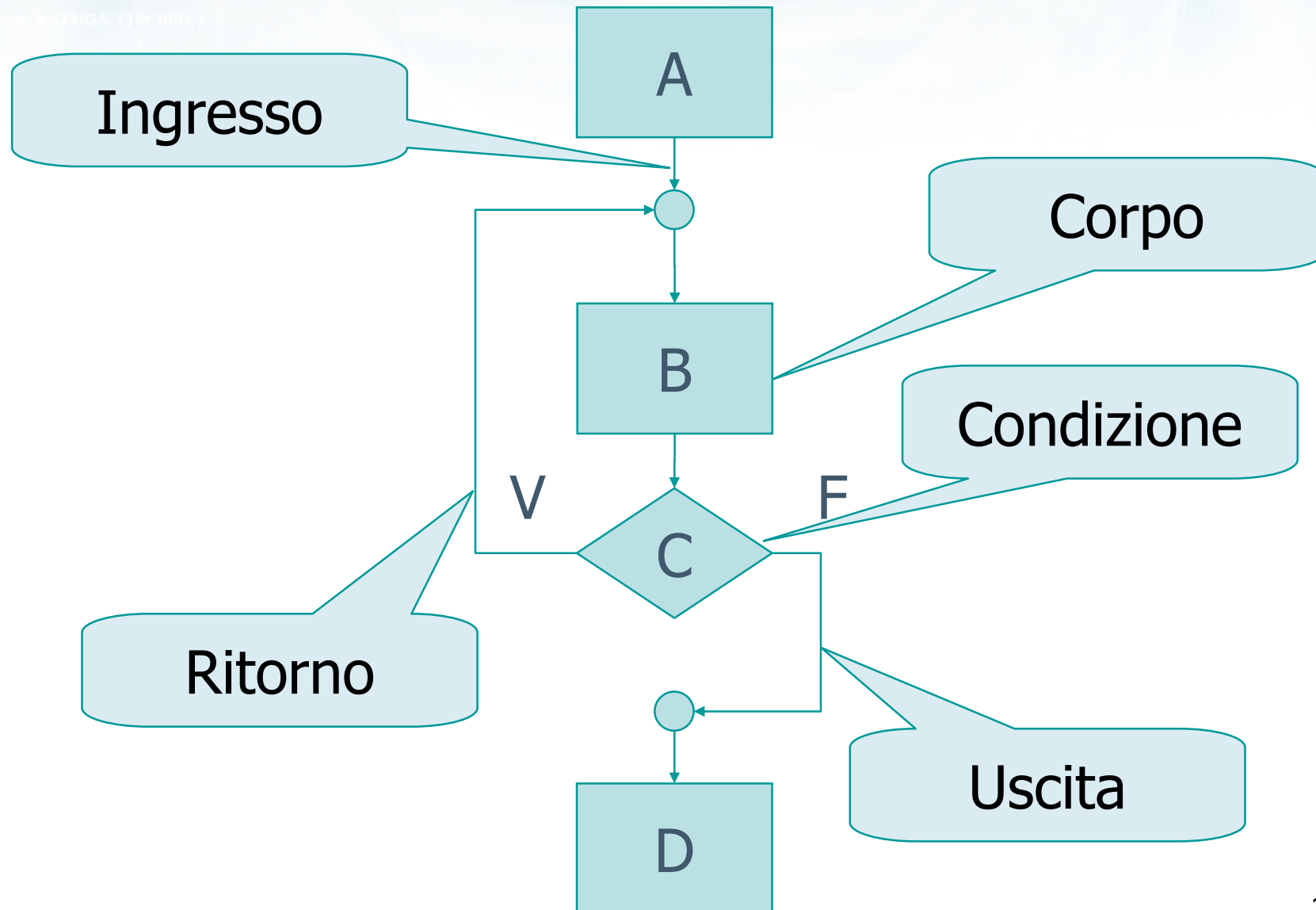


A
C → Vero
B
C → Vero
B
C → Falso
D

Notazione grafica (do-while)

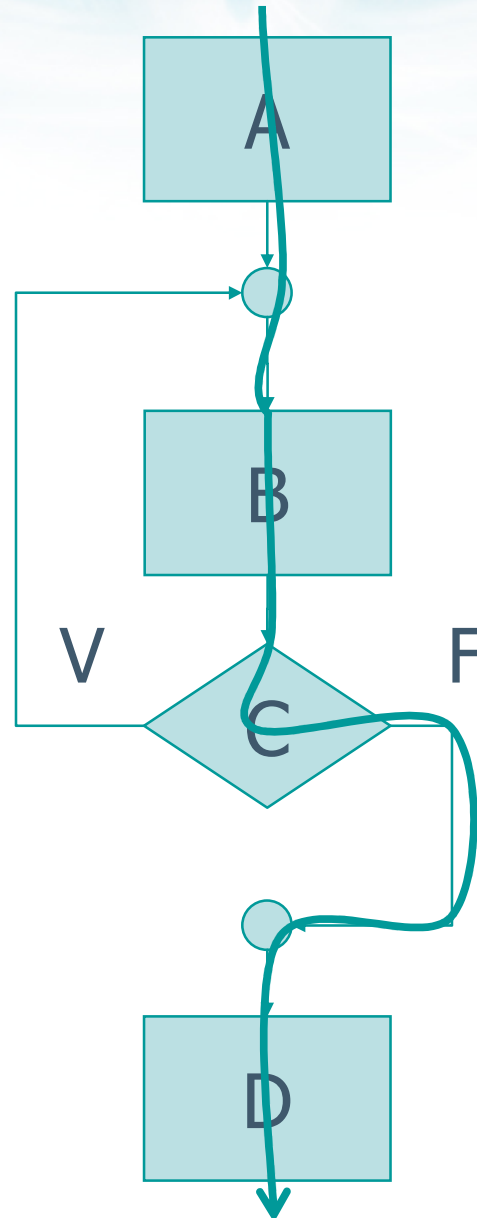


Notazione grafica (do-while)



Flussi di esecuzione

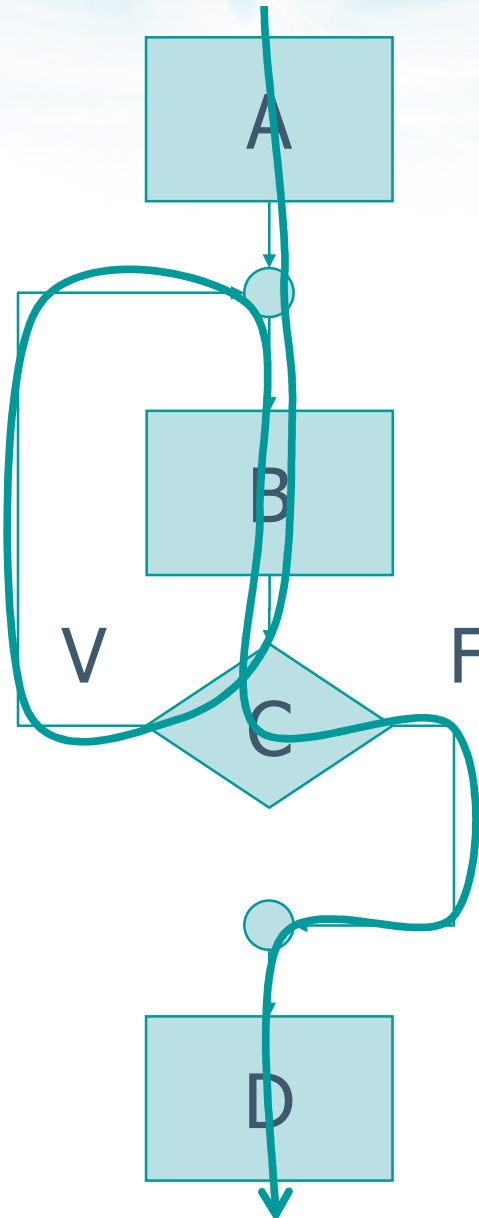
Una iterazione



A
B
C → Falso
D

Flussi di esecuzione

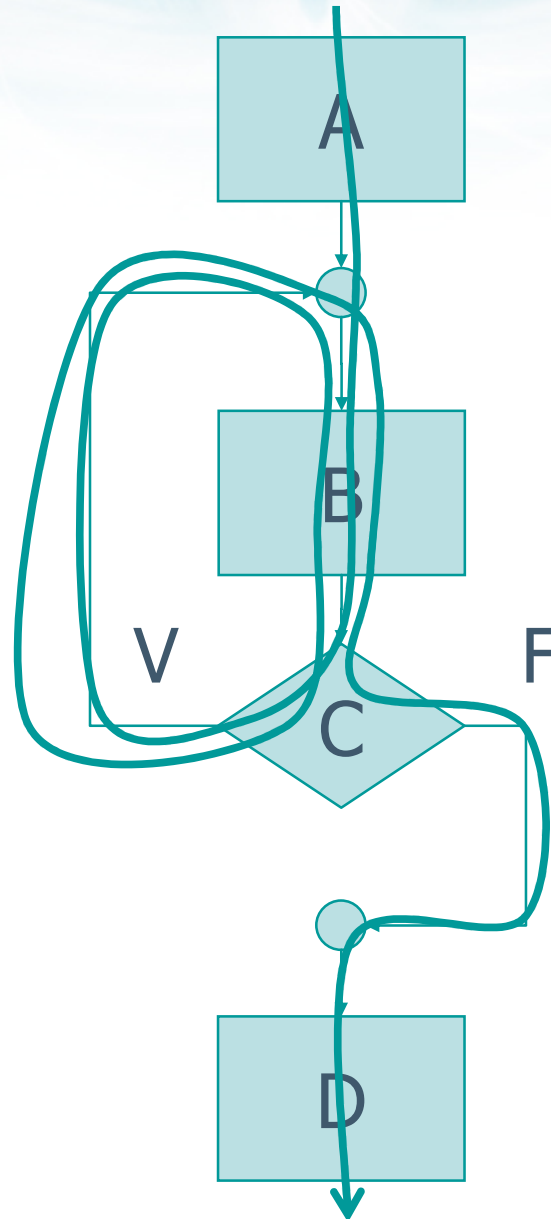
Due iterazioni



A
B
C → Vero
B
C → Falso
D

Flussi di esecuzione

Tre iterazioni



A
B
C → Vero
B
C → Vero
B
C → Falso
D

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



La ripetizione

Struttura di un ciclo

- Nello strutturare un ciclo occorre garantire:
 - Che il ciclo possa terminare
 - Che il numero di iterazioni sia quello desiderato
- Il corpo centrale del ciclo può venire eseguito più volte:
 - La prima volta lavorerà con variabili che sono state inizializzate al di fuori del ciclo
 - Le volte successive lavorerà con variabili che possono essere state modificate nell'iterazione precedente
 - Garantire la correttezza sia della prima, che delle altre iterazioni

Anatomia di un ciclo (1/5)

- Conviene concepire il ciclo come 4 fasi
 - Inizializzazione
 - Condizione di ripetizione
 - Corpo
 - Aggiornamento

Anatomia di un ciclo (2/5)

➤ Conviene concepire il ciclo come 4 fasi

- Inizializzazione

- Assegnazione del valore iniziale a tutte le variabili che vengono lette durante il ciclo (nel corpo o nella condizione)

- Condizione di ripetizione

- Corpo

- Aggiornamento

Anatomia di un ciclo (3/5)

➤ Conviene concepire il ciclo come 4 fasi

- Inizializzazione
- Condizione di ripetizione
 - Condizione, di solito inizialmente vera, che al termine del ciclo diventerà falsa
 - Deve dipendere da variabili che saranno modificate all'interno del ciclo (nel corpo o nell'aggiornamento)
- Corpo
- Aggiornamento

Anatomia di un ciclo (4/5)

- **Conviene concepire il ciclo come 4 fasi**
 - **Inizializzazione**
 - **Condizione di ripetizione**
 - **Corpo**
 - Le istruzioni che effettivamente occorre ripetere
 - Sono lo scopo per cui il ciclo viene realizzato
 - Posso usare le variabili inizializzate
 - Posso modificare le variabili
 - **Aggiornamento**

Anatomia di un ciclo (5/5)

➤ Conviene concepire il ciclo come 4 fasi

- Inizializzazione
- Condizione di ripetizione
- Corpo
- Aggiornamento
 - Modifica di una o più variabili in grado di aggiornare il valore della condizione di ripetizione
 - Tengono "traccia" del progresso dell'iterazione

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



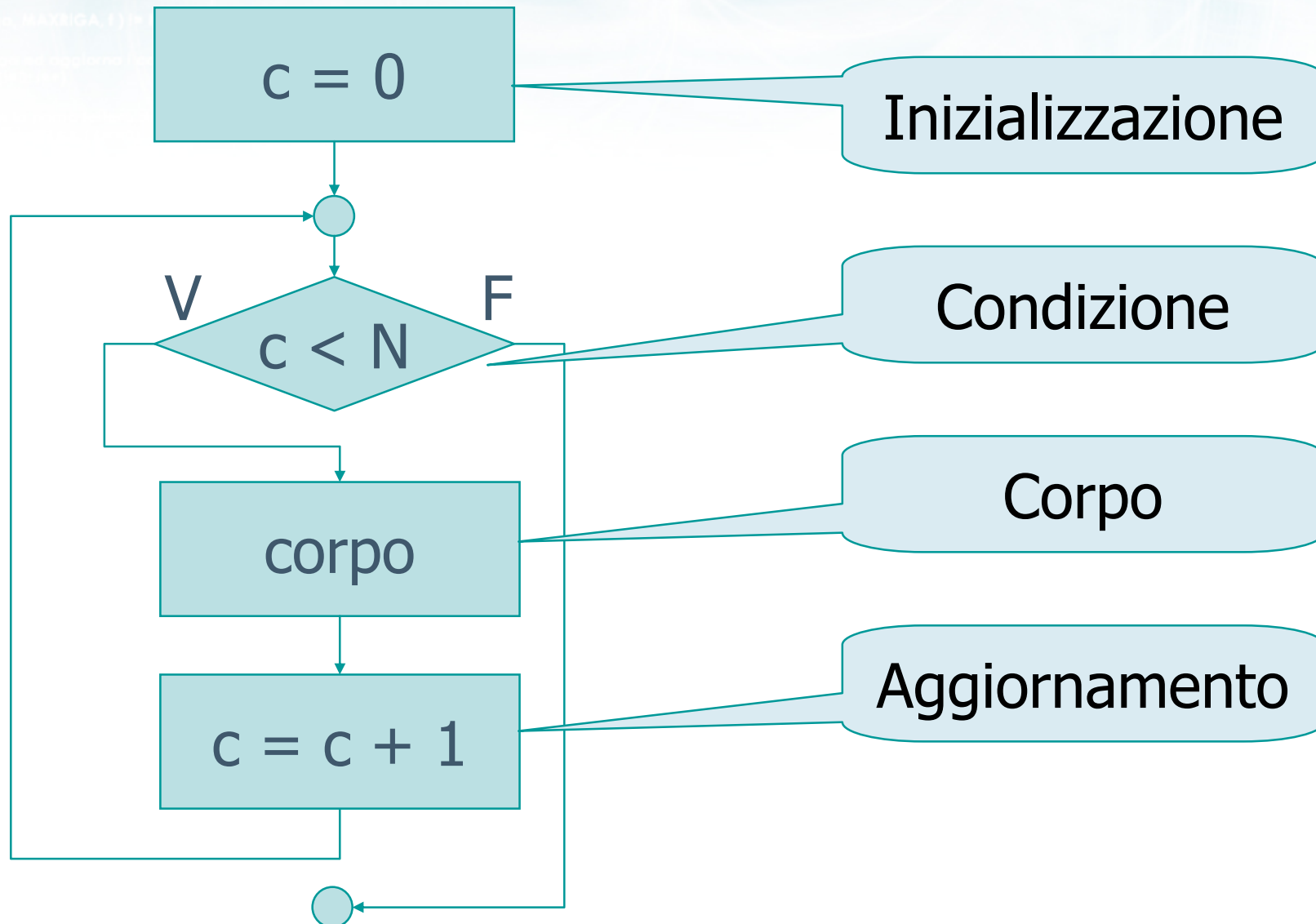
La ripetizione

Numero di iterazioni note

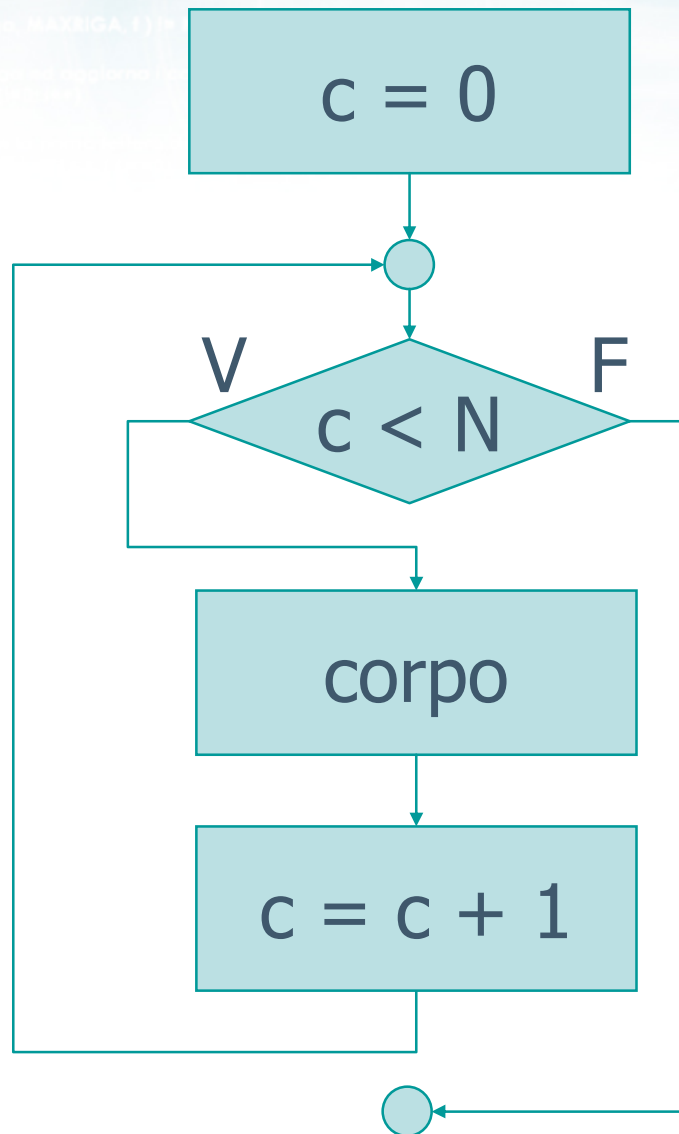
Tipologie di cicli

- Cicli in cui il numero di iterazioni sia noto a priori, ossia prima di entrare nel ciclo stesso
 - Solitamente si usa una variabile "contatore"
 - L'aggiornamento consiste in un incremento o decremento della variabile
- Cicli in cui il numero di iterazioni non sia noto a priori, ma dipenda dai dati elaborati nel ciclo
 - Solitamente si usa una condizione dipendente da una variabile letta da tastiera oppure calcolata nel corpo del ciclo
 - Difficile distinguere il corpo dall'aggiornamento
 - Problema di inizializzazione

Cicli con iterazioni note

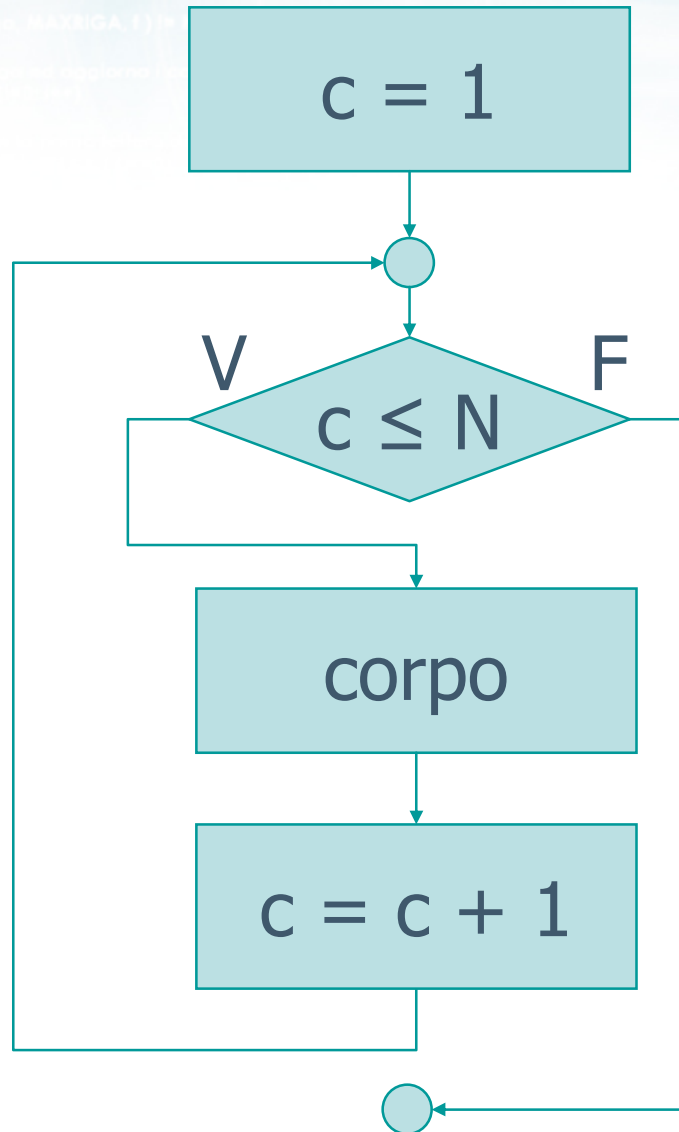


Cicli con iterazioni note



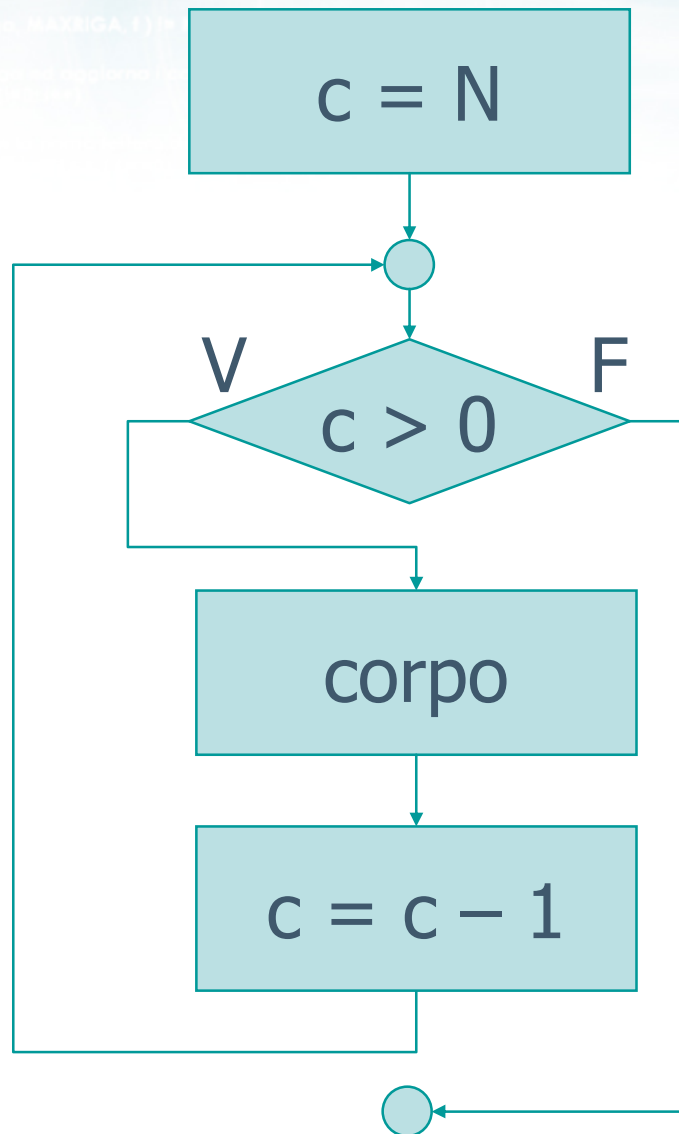
- Forma $0 \dots N-1$
- Prima iterazione:
 - $c=0$
- Ultima iterazione:
 - $c=N-1$
- Corpo ripetuto:
 - N volte
- Al termine del ciclo
 - $c=N$
 - condizione falsa

Cicli con iterazioni note



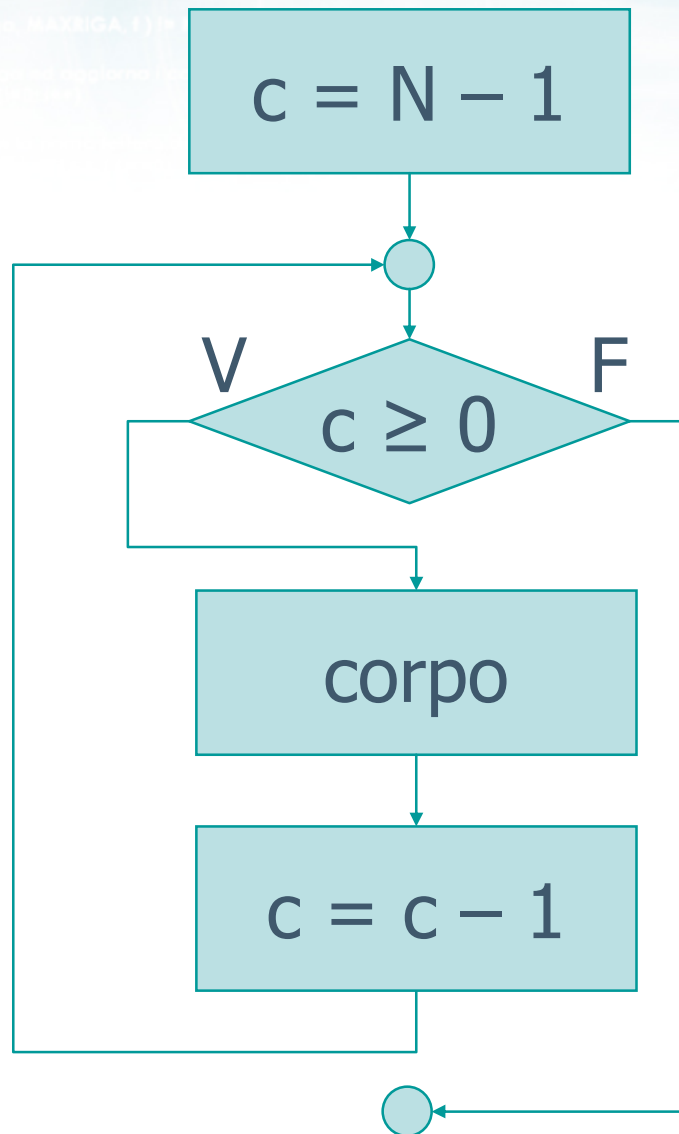
- Forma 1...N
- Prima iterazione:
 - $c=1$
- Ultima iterazione:
 - $c=N$
- Corpo ripetuto:
 - N volte
- Al termine del ciclo
 - $c=N+1$
 - condizione falsa

Cicli con iterazioni note



- Forma N...1
- Prima iterazione:
 - $c=N$
- Ultima iterazione:
 - $c=1$
- Corpo ripetuto:
 - N volte
- Al termine del ciclo
 - $c=0$
 - condizione falsa

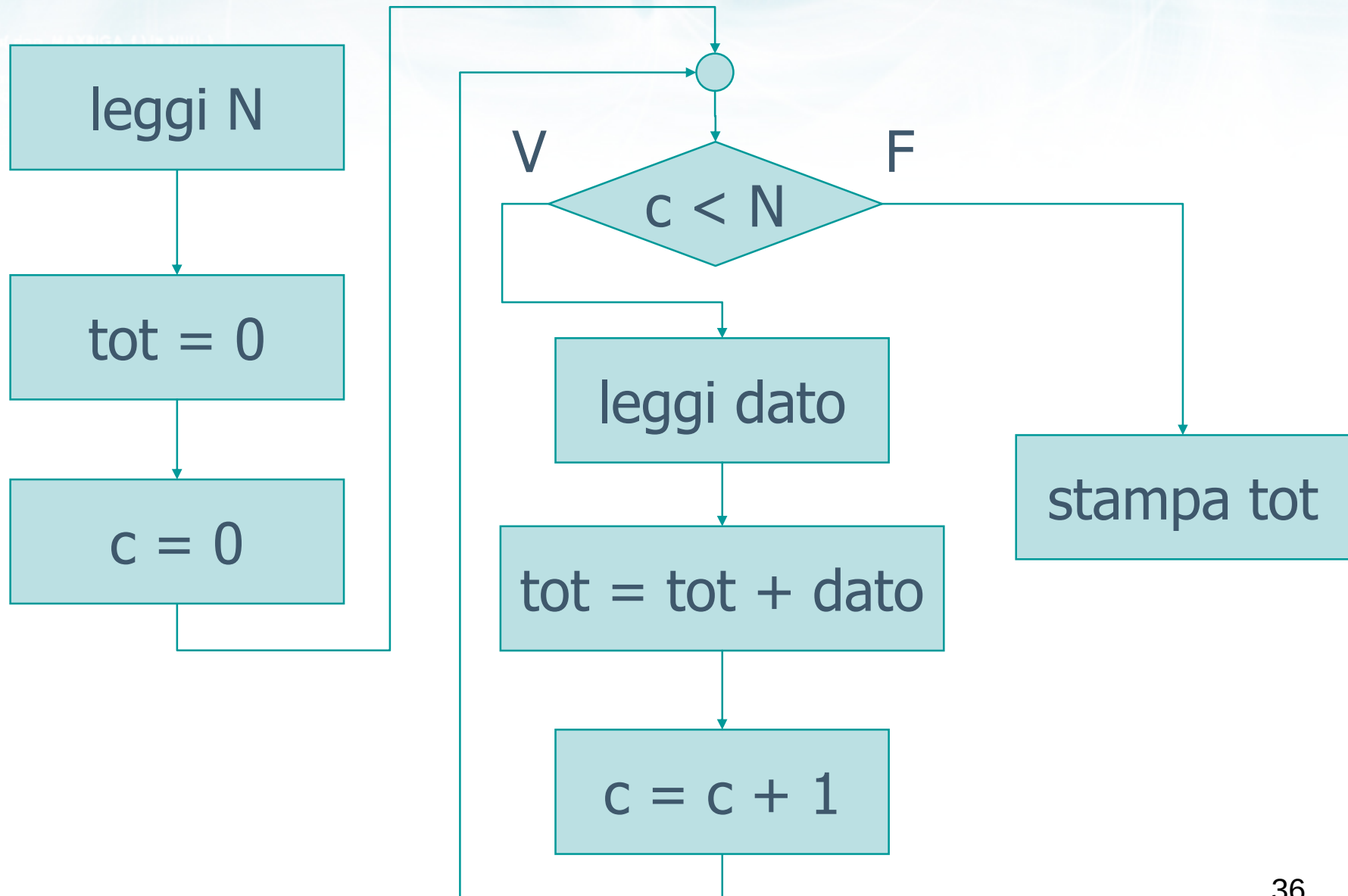
Cicli con iterazioni note



- Forma $N-1 \dots 0$
- Prima iterazione:
 - $c = N-1$
- Ultima iterazione:
 - $c = 0$
- Corpo ripetuto:
 - N volte
- Al termine del ciclo
 - $c = -1$
 - condizione falsa

- Acquisire da tastiera una sequenza di numeri interi e stamparne la somma.
- Il programma
 - inizialmente chiede all'utente quanti numeri intende inserire
 - in seguito richiede uno ad uno i dati
 - infine stampa la somma

Soluzione



```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



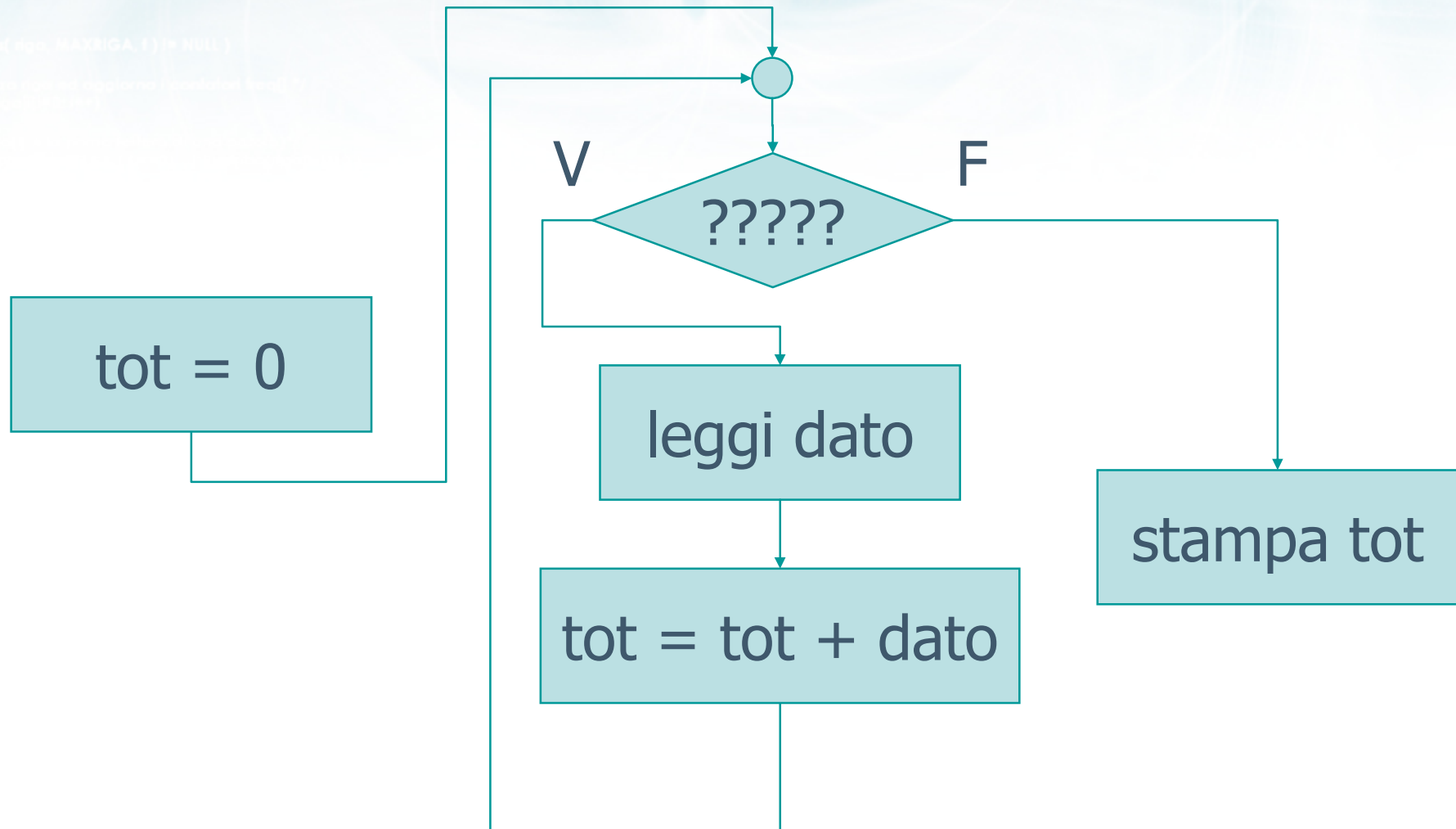
La ripetizione

Numero di iterazioni ignote

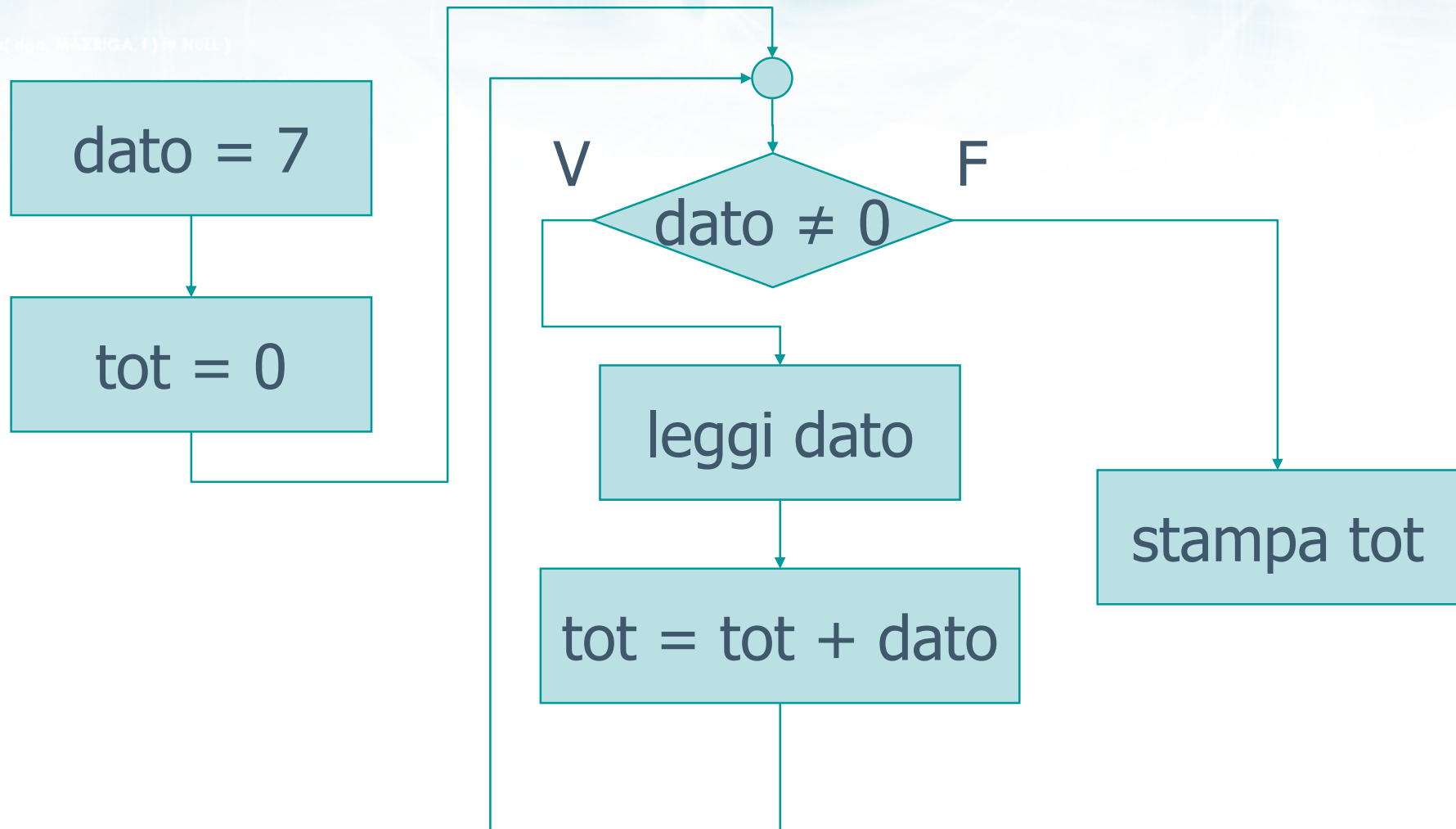
Cicli con iterazioni ignote

- Non esiste uno schema generale
- Esempio:
 - Acquisire da tastiera una sequenza di numeri interi e stamparne la somma.
 - Il termine della sequenza viene indicato inserendo un dato pari a zero.

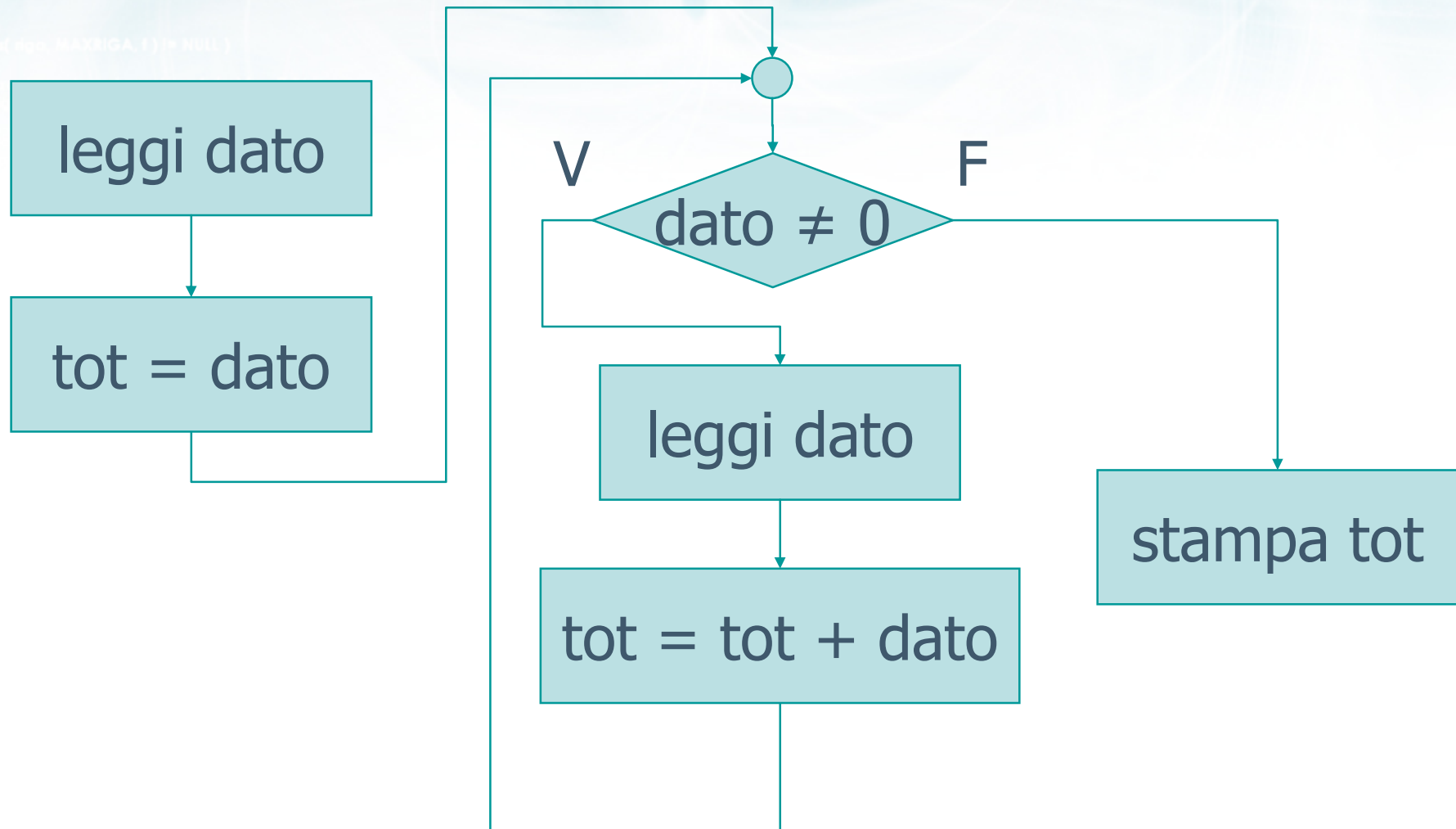
Soluzione parziale



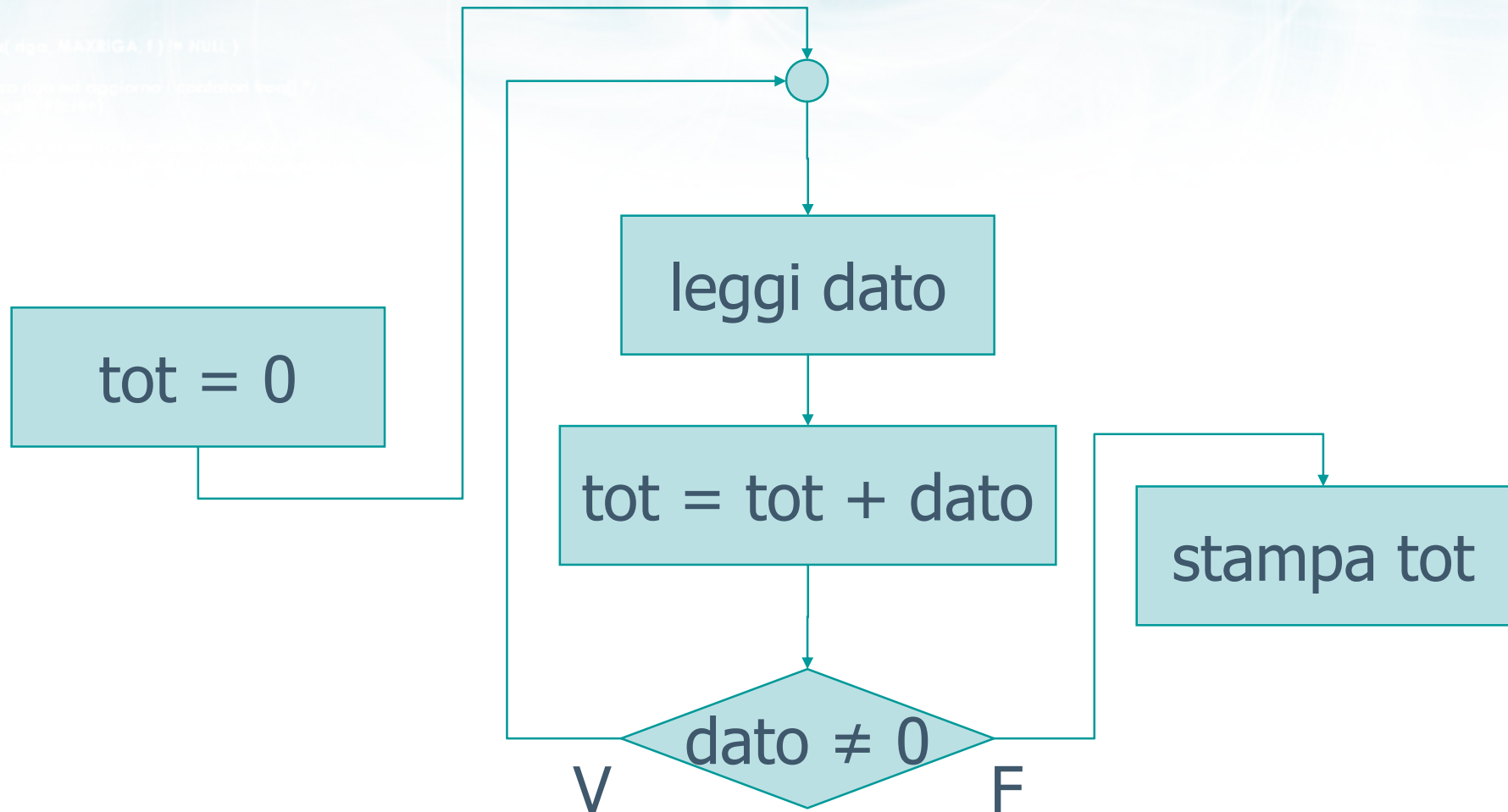
Soluzione 1



Soluzione 2



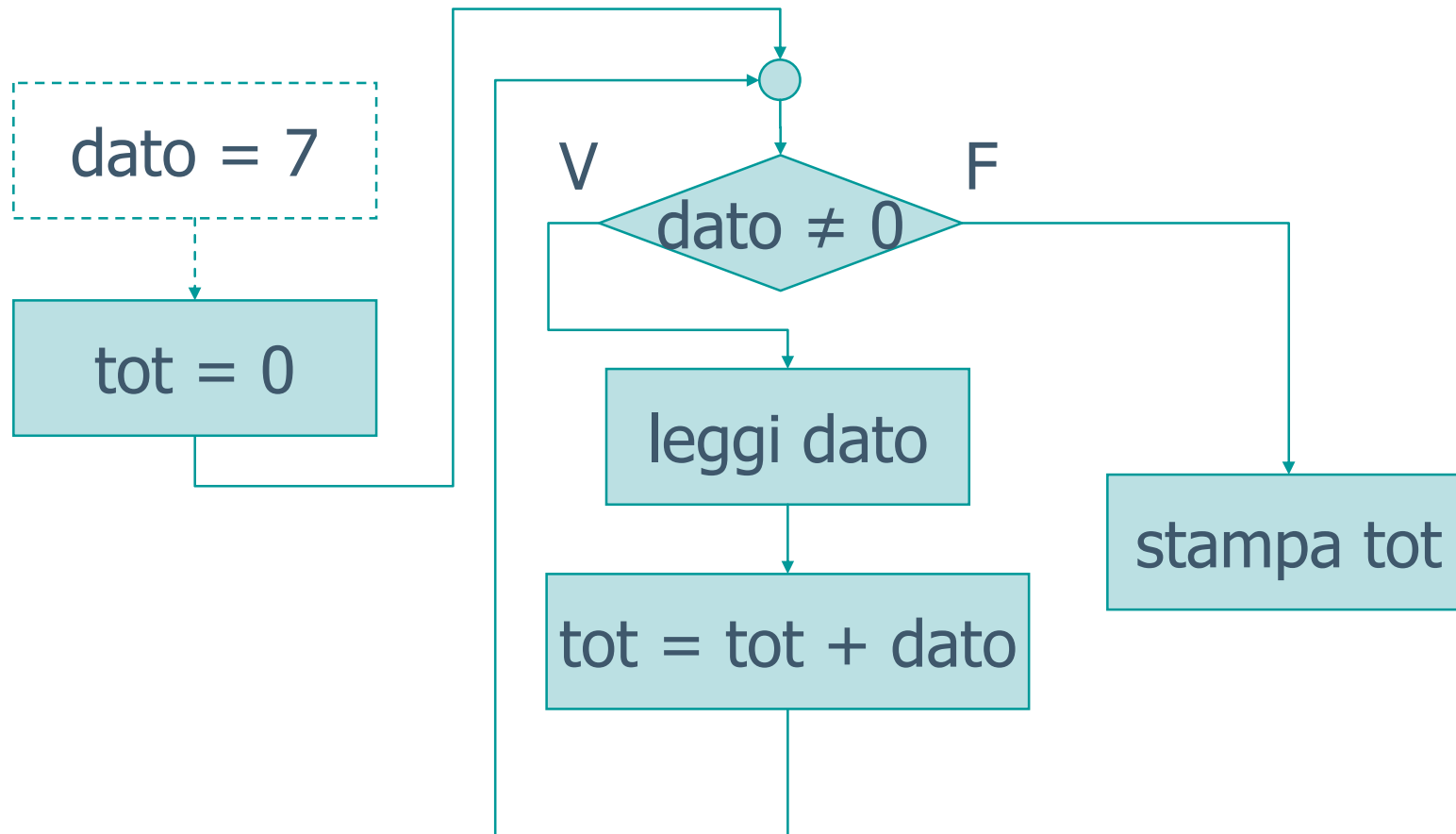
Soluzione 3





Errore frequente

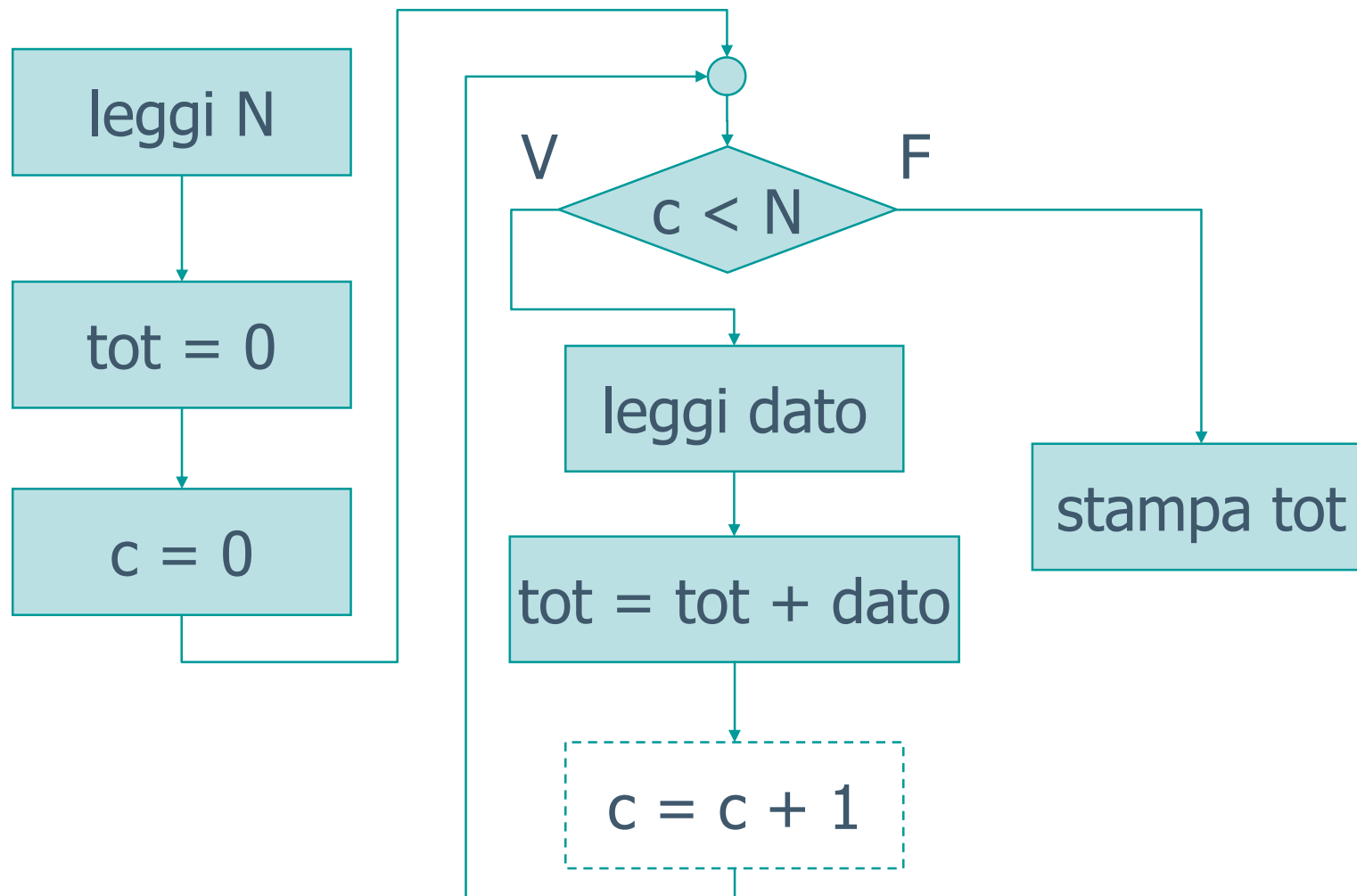
- Dimenticare l'inizializzazione di una variabile utilizzata all'interno del ciclo





Errore frequente

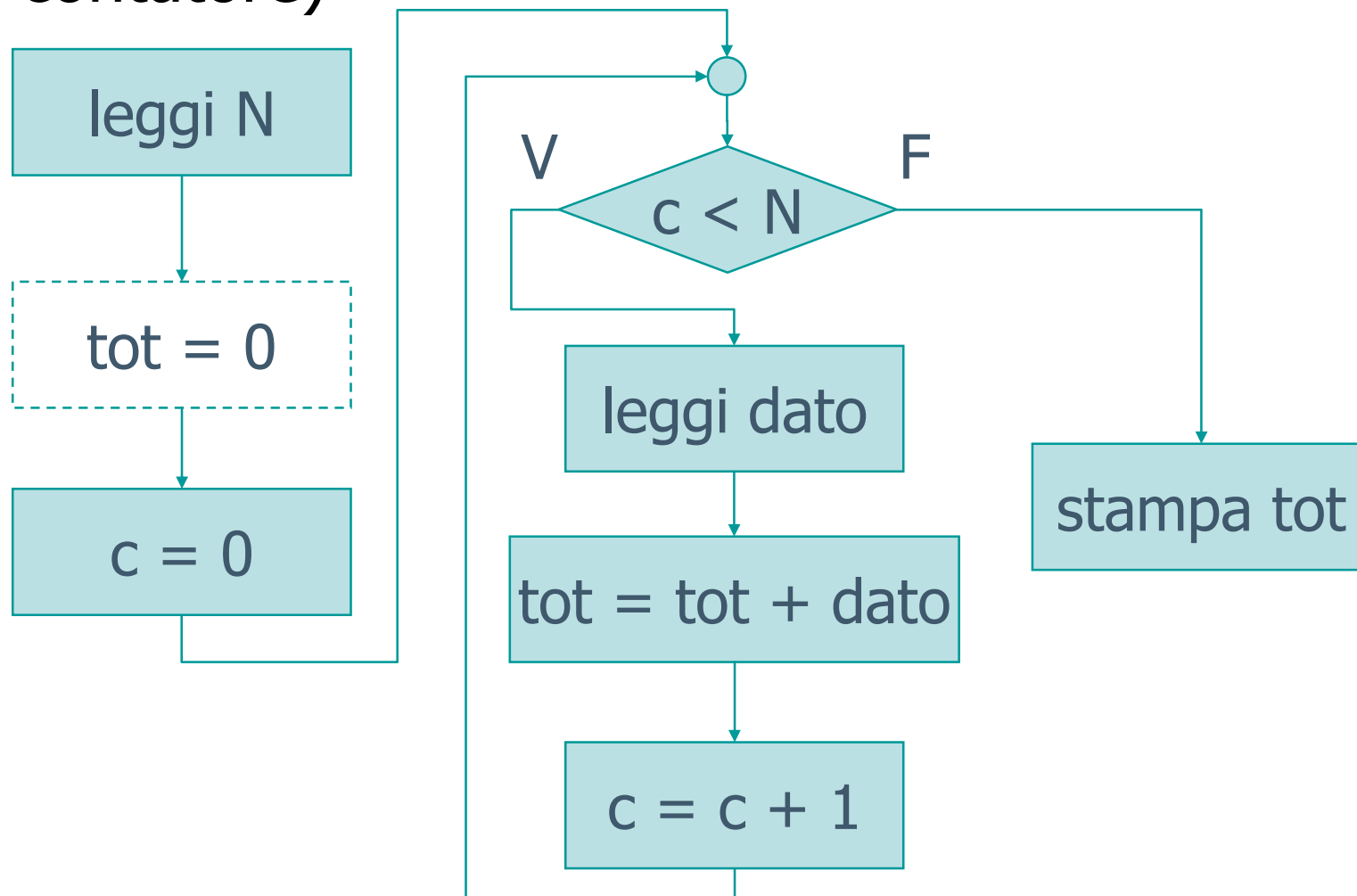
- Dimenticare l'incremento della variabile contatore





Errore frequente

- Dimenticare di inizializzare le altre variabili (oltre al contatore)




```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
```

```
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
```

```
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
```

```
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Istruzione while

Istruzione while

- Sintassi dell'istruzione
- Esercizio "Media aritmetica"
- Esecuzione del programma
- Cicli while e annidati
- Esercizio "Quadrato"

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
```

```
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
```

```
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Istruzione while

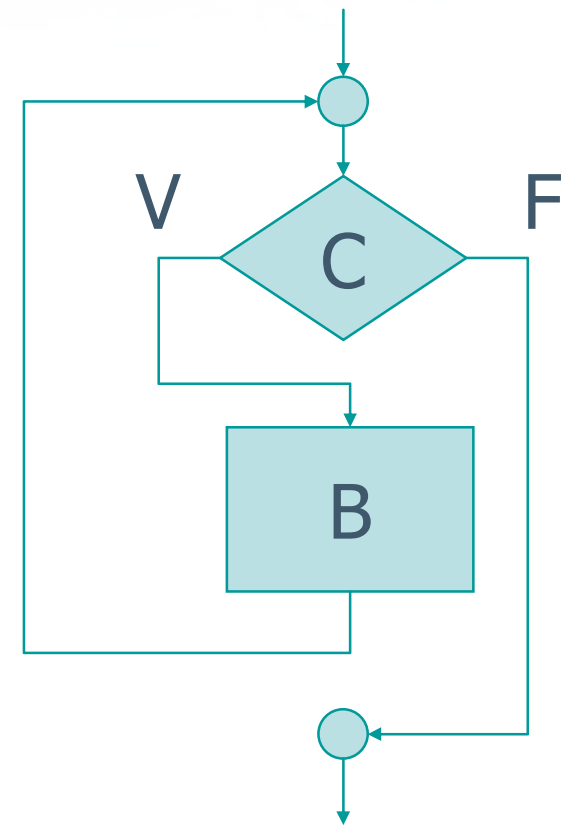
Sintassi dell'istruzione

Istruzioni di ripetizione in C

- Nel linguaggio C esistono tre distinte istruzioni di iterazione
 - `while`
 - `do-while`
 - `for`
- La forma più generale è l'istruzione di tipo `while`
- L'istruzione `do-while` si usa in taluni contesti (es. controllo errori di input)
- L'istruzione `for` è usatissima, soprattutto per numero di iterazioni noto

Istruzione while

```
while ( C )  
{  
    B ;  
}
```



Comportamento del while

```
while ( C )  
{  
    B ;  
}
```

1. Valuta la condizione **C**
2. Se **C** è falsa, salta completamente l'iterazione e vai all'istruzione che segue la }
3. Se **C** è vera, esegui una volta il blocco di istruzioni **B**
4. Al termine del blocco **B**, ritorna al punto 1. per rivalutare la condizione **C**

Numero di iterazioni note

```
int i, N ;
```

```
i = 0 ;
```

```
while ( i < N )
```

```
{
```

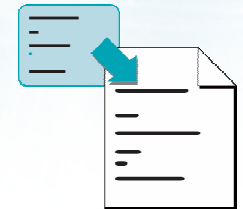
```
    /* Corpo dell'iterazione */
```

```
    ...
```

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

```
}
```


Esempio



num1-10.c

```
int i ;

i = 1 ;
while ( i <= 10 )
{
    printf("Numero = %d\n", i) ;
    i = i + 1 ;
}
```


Esempio



fatt.c

```
int i, n ;
float f ;
.... /* leggi n */ ....
i = 2 ;
f = 1.0 ;
while ( i <= n )
{
    f = f * i ;
    i = i + 1 ;
}
printf("Fattoriale di %d = %f\n",
      n, f);
```

Particolarità

- Nel caso in cui il corpo del `while` sia composto di una sola istruzione, si possono omettere le parentesi graffe
 - Non succede quasi mai

```
while ( C )  
{  
    B ;  
}
```



```
while ( C )  
    B ;
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```

Istruzione while

Esercizio “Media aritmetica”

Esercizio “Media aritmetica”

- Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Leggere n numeri reali
 - Calcolare e visualizzare la media aritmetica di tali numeri
- Osservazione
 - Attenzione al caso in cui $n \leq 0$

Analisi

C:\> Prompt dei comandi

MEDIA ARITMETICA

Introduci n: 3

Ora introduci 3 valori

valore 1: 6.5

valore 2: 2.5

valore 3: 3.0

Risultato: 4.000000

Algoritmo

- Acquisisci n
- Inizializza totale = 0
- Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
- Calcola e stampa la media = totale / n

- Acquisisci n
- Se $n > 0$
 - Inizializza totale = 0
 - Ripeti n volte
 - Acquisisci un dato
 - Somma il dato al totale dei dati acquisiti
 - Calcola e stampa la media = totale / n
- Altrimenti stampa messaggio di errore

Traduzione in C (1/3)

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

```
int main(void)
{
    int i, n ;
    float dato ;
    float somma ;

    printf("MEDIA ARITMETICA\n");

    /* Leggi n */
    printf("Introduci n: ");
    scanf("%d", &n) ;
```



media.c

Traduzione in C (2/3)

```
/* Controlla la correttezza del valore n */
```

```
if( n>0 )  
{
```

```
    /* n corretto... procedi! */
```

```
    ....vedi lucido seguente....
```

```
}  
else  
{
```

```
    /* n errato in quanto e' n <= 0 */
```

```
    printf("Non ci sono dati da inserire\n");
```

```
    printf("Impossibile calcolare la media\n");
```

```
}
```

```
} /* main */
```

media.c

Traduzione in C (3/3)

```
/* Leggi i valori e calcola la media */  
printf("Ora immetti %d valori\n", n) ;
```

```
    somma = 0.0 ;  
    i = 0 ;  
    while( i < n )  
    {  
        printf("valore %d: ", i+1) ;  
        scanf("%f", &dato) ;  
  
        somma = somma + dato ;  
  
        i = i + 1 ;  
    }
```

```
printf("Risultato: %f\n", somma/n) ;
```



media.c

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

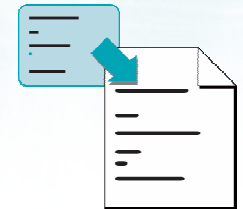
```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Istruzione while

Esecuzione del programma

Verifica "Media aritmetica"



media.c

```
C:\ D:\home\mirror\CDROM\materiale\U3C\media.exe
MEDIA ARITMETICA
Introduci n: 3
Ora immetti 3 valori
Valore 1: 6.5
Valore 2: 3.2
Valore 3: 1.3
Risultato: 3.666667
Premere un tasto per continuare . . .
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



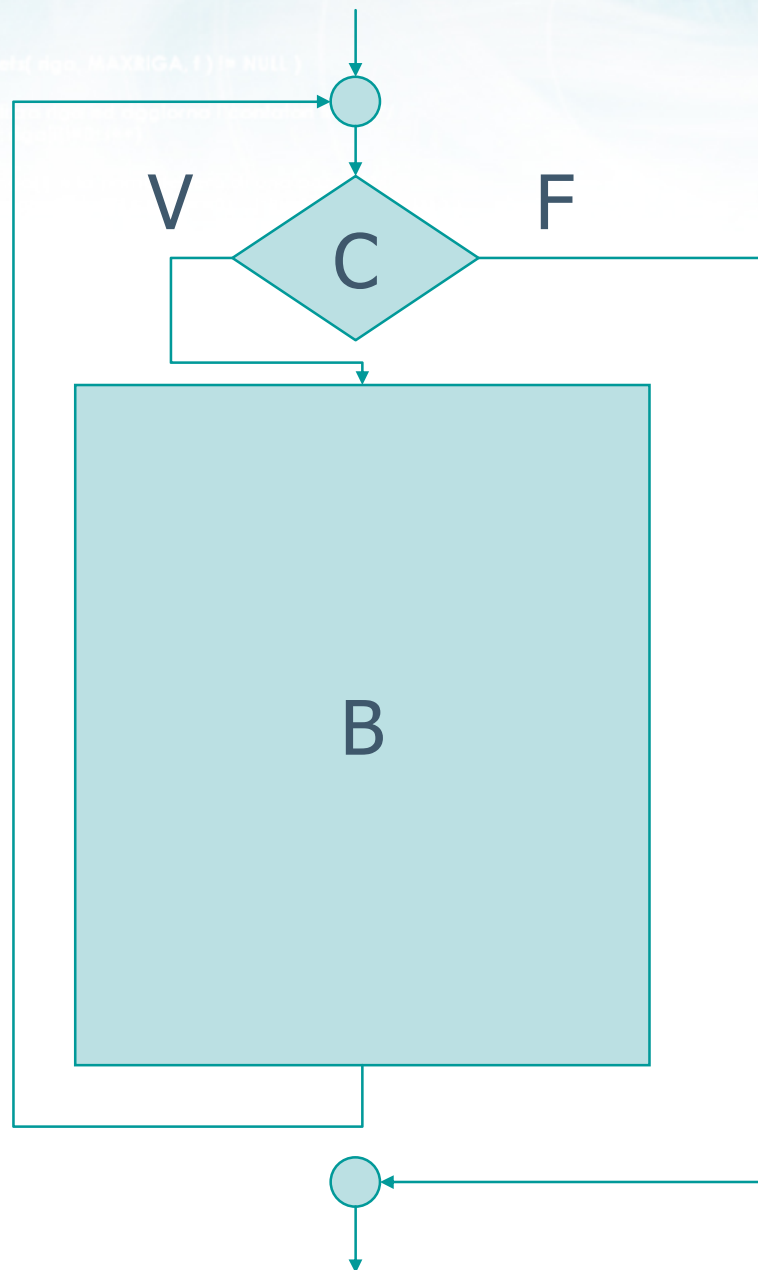
Istruzione while

Cicli while annidati

Annidamento di cicli

- All'interno del corpo del ciclo `while` è possibile racchiudere qualsiasi altra istruzione `C`
- In particolare, è possibile racchiudere un'istruzione `while` all'interno di un'altra istruzione `while`
- In tal caso, per ogni singola iterazione del ciclo `while` più esterno, vi saranno tutte le iterazioni previste per il ciclo più interno

Cicli while annidati

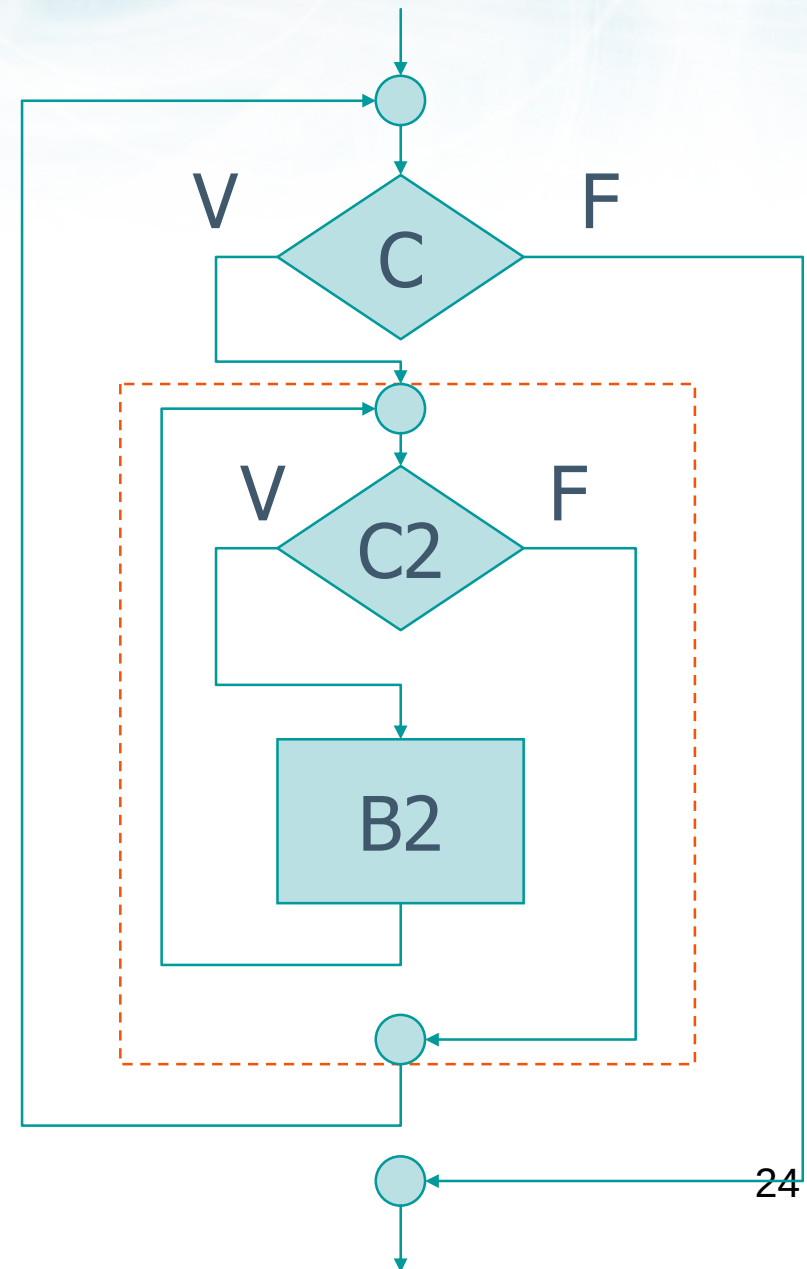
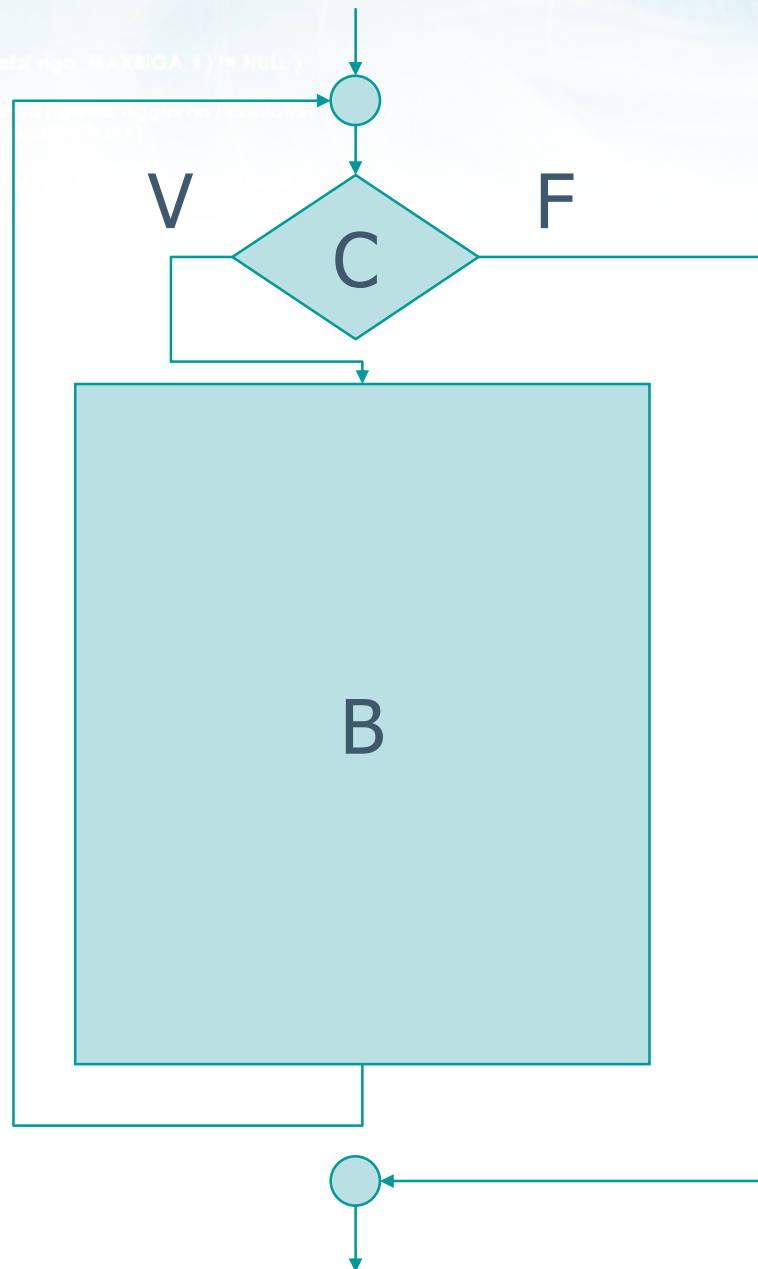



```

if(argc != 2)
{
    fprintf(stderr, "TFSOss: serve un parametro con il nome del file\n");
    exit(1);
}
f = fopen(argv[1], "r");
if(f == NULL)
{
    fprintf(stderr, "TFSOss: file non trovato\n");
    exit(1);
}

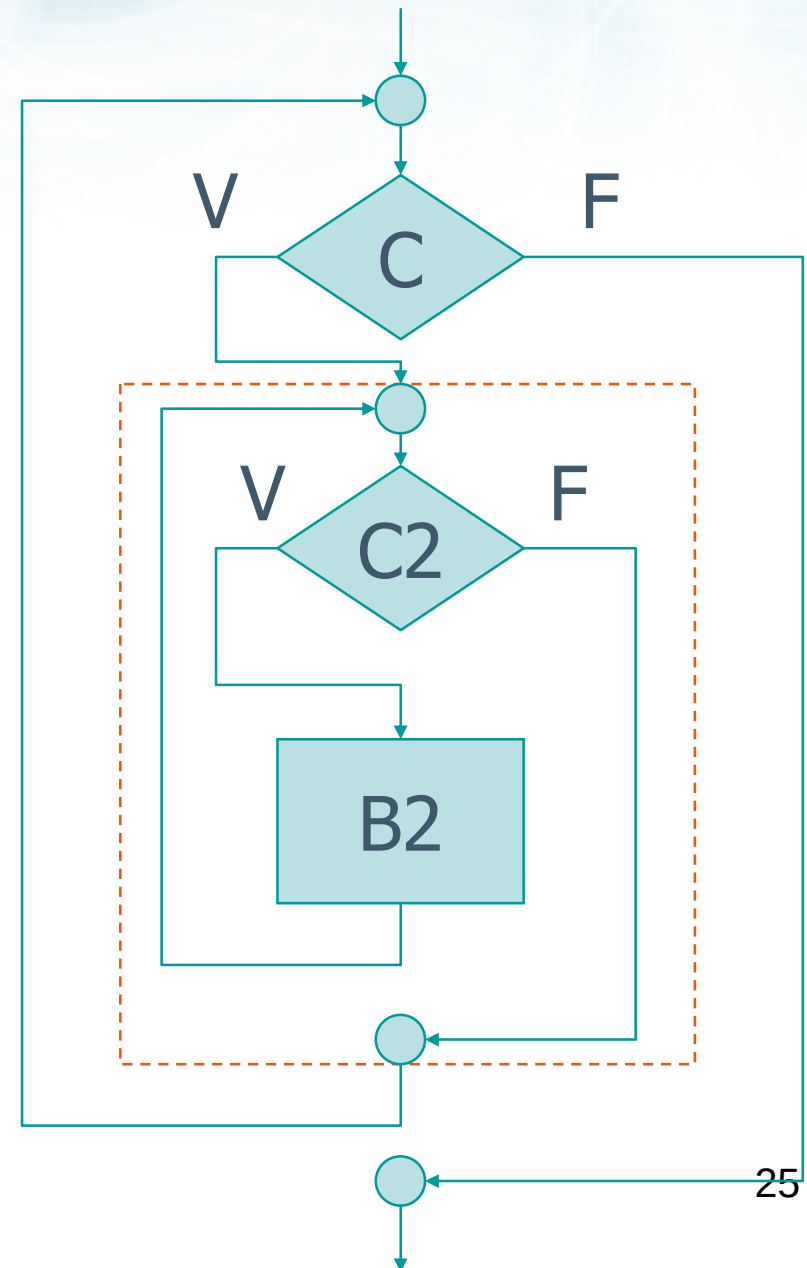
```

Cicli while e annidati



Cicli while e annidati

```
while( c )  
{  
    while( c2 )  
    {  
        B2 ;  
    }  
}
```



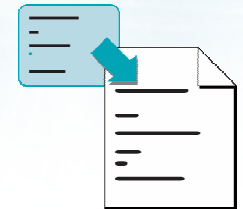
Esempio

```
i = 0 ;  
while( i < N )  
{  
    j = 0 ;  
    while( j < N )  
    {  
        printf("i=%d - j=%d\n", i, j);  
  
        j = j + 1 ;  
    }  
  
    i = i + 1 ;  
}
```



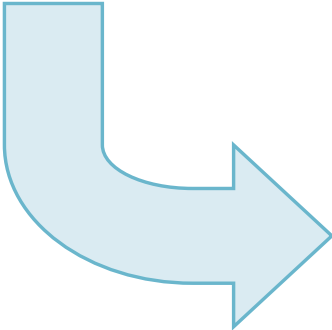
conta99.c

Esempio



conta99.c

```
i = 0 ;  
while( i < N )  
{  
    j = 0 ;  
    while( j < N )  
    {  
        printf("i=%d - j=%d\n", i, j);  
        j = j + 1 ;  
    }  
    i = i + 1 ;  
}
```



```
i=0 - j=0  
i=0 - j=1  
i=0 - j=2  
i=1 - j=0  
i=1 - j=1  
i=1 - j=2  
i=2 - j=0  
i=2 - j=1  
i=2 - j=2
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Istruzione while

Esercizio "Quadrato"

Esercizio "Quadrato"

- Si realizzi un programma C in grado di
 - Leggere un numero naturale n
 - Visualizzare un quadrato di lato n costituito da asterischi

Analisi

C:\> Prompt dei comandi

QUADRATO

Introduci n: 5

Algoritmo

- Acquisisci n
- Ripeti n volte
 - Stampa una riga di n asterischi

Algoritmo

- Acquisisci n
- Ripeti n volte

- Stampa una riga di n asterischi

- Ripeti n volte
 - Stampa un singolo asterisco
- Vai a capo

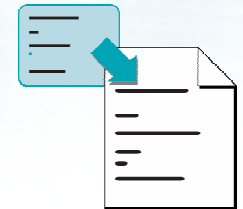
Traduzione in C



quadrato.c

```
i = 0 ;  
while( i < n )  
{  
    j = 0 ;  
    while( j < n )  
    {  
        printf("*") ;  
        j = j + 1 ;  
    }  
  
    printf("\n");  
  
    i = i + 1 ;  
}
```

Traduzione in C



quadrato.c

```
i = 0 ;  
while( i < n )  
{  
    j = 0 ;  
    while( j < n )  
    {  
        printf("*") ;  
        j = j + 1 ;  
    }  
    printf("\n");  
    i = i + 1 ;  
}
```

Ripeti n volte

Stampa una riga
di n asterischi

Traduzione in C



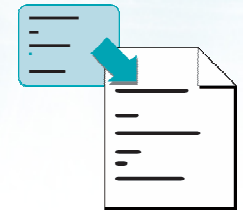
quadrato.c

```
i = 0 ;  
while( i < n )  
{  
    j = 0 ;  
    while( j < n )  
    {  
        printf("*") ;  
        j = j + 1 ;  
    }  
    printf("\n");  
    i = i + 1 ;  
}
```

Stampa n
asterischi

Vai a capo

Traduzione in C



quadrato.c

```
i = 0 ;  
while( i < n )  
{  
    j = 0 ;  
    while( j < n )  
    {  
        printf("*") ;  
        j = j + 1 ;  
    }  
    printf("\n");  
    i = i + 1 ;  
}
```

Ripeti n volte

Stampa un
asterisco

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Schemi ricorrenti nei cicli

Schemi ricorrenti nei cicli

- Contatori
- Accumulatori
- Flag
- Esistenza e universalità

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
                           delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Schemi ricorrenti nei cicli

Contatori

- Spesso in un ciclo è utile sapere
 - Quante iterazioni sono state fatte
 - Quante iterazioni rimangono da fare
 - Quale numero di iterazione sia quella corrente
- Per questi scopi si usano delle “normali” variabili intere, dette **contatori**
 - Inizializzate prima del ciclo
 - Incrementate/decrementate ad ogni iterazione
 - Oppure incrementate/decrementate ogni volta che si riscontra una certa condizione

Contatori

```
int i, N ;
```

```
...  
[ i = 0 ; ]
```

```
while ( i < N )
```

```
{
```

```
    printf("Iterazione %d\n", i+1) ;
```

```
    [ i = i + 1 ; ]
```

```
}
```

- Scrivere un programma in C che
 - legga dall'utente 10 numeri interi
 - al termine dell'inserimento, stampi
 - quanti tra i numeri inseriti sono positivi
 - quanti tra i numeri inseriti sono negativi
 - quanti tra i numeri inseriti sono nulli

Soluzione (1/3)



contaposneg.c

```
int npos, nneg, nzero ;
```

```
....
```

```
npos = 0 ;
```

```
nneg = 0 ;
```

```
nzero = 0 ;
```

Soluzione (2/3)

```
i = 0 ;
while( i < n )
{
    printf("Inserisci dato %d: ", i+1);
    scanf("%d", &dato);

    if( dato > 0 )
        npos = npos + 1 ;
    else if( dato < 0 )
        nneg = nneg + 1 ;
    else
        nzero = nzero + 1 ;

    i = i + 1 ;
}
```

contaposneg.c

Soluzione (3/3)



contaposneg.c

```
printf("Numeri positivi: %d\n", npos);  
printf("Numeri negativi: %d\n", nneg);  
printf("Numeri nulli: %d\n", nzero);
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```

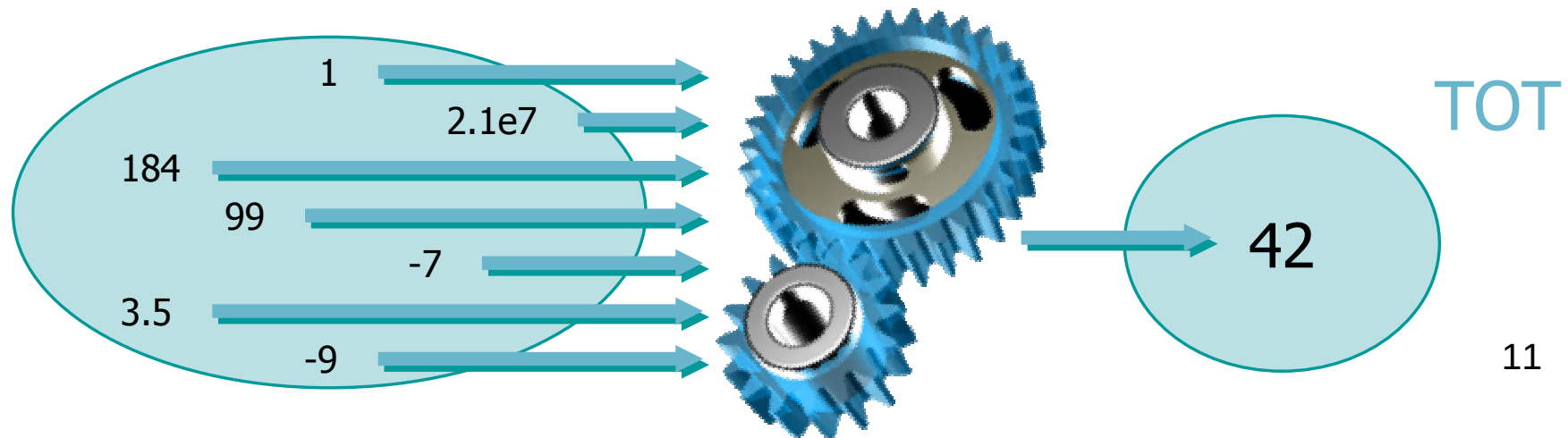


Schemi ricorrenti nei cicli

Accumulatori

Accumulatori (1/2)

- Spesso in un ciclo occorre calcolare un valore TOT che dipende dall'insieme dei valori analizzati nelle singole iterazioni
- Esempi:
 - TOT = sommatoria dei dati analizzati
 - TOT = produttoria dei dati analizzati
 - TOT = massimo, minimo dei dati analizzati



Accumulatori (2/2)

➤ In questo caso si usano delle variabili (interi o reali) dette **accumulatori**

- Inizializzare TOT al valore che dovrebbe avere in assenza di dati (come se fosse $n=0$)
- Ad ogni iterazione, aggiornare TOT tenendo conto del dato appena analizzato
- Al termine del ciclo, TOT avrà il valore desiderato

Esempio: somma primi 10 interi

- Si scriva un programma in C che stampi il valore della somma dei primi 10 numeri interi

➤ Inizializzazione di TOT

- Qual è la somma dei primi 0 numeri interi?
- $TOT = 0$

➤ Aggiornamento di TOT

- Sapendo che TOT è la somma dei primi (i-1) numeri interi, e sapendo che il prossimo numero intero da sommare vale i, quanto dovrà valere TOT?
- $TOT = TOT + i$

Soluzione: somma primi 10 interi

```
int tot ;

i = 1 ;
tot = 0 ;
while( i<=10 )
{
    tot = tot + i ;

    i = i + 1 ;
}

printf("La somma dei numeri da 1 a 10 ");
printf("vale %d\n", tot) ;
```

Esempio: fattoriale di K

- Si scriva un programma in C che, dato un numero intero K, calcoli e stampi il fattoriale di K
- $TOT = K!$

$$TOT = K! = \prod_{i=1}^{i=K} i$$

➤ Inizializzazione di TOT

- Qual è il valore del fattoriale per $K=0$?
- $TOT = 1.0$

➤ Aggiornamento di TOT

- Sapendo che TOT è pari al fattoriale di $i-1$, e sapendo che il prossimo numero da considerare è i , quanto dovrà valere TOT?
- $TOT = TOT * i$

Soluzione: fattoriale di K

```
float tot ;

i = 1 ;
tot = 1.0 ;
while( i<=K )
{
    tot = tot * i ;

    i = i + 1 ;
}

printf("Il fattoriale di %d ", K);
printf("vale %f\n", tot) ;
```

Esempio: massimo

- Si scriva un programma in C che
 - acquisisca da tastiera N numeri reali
 - stampi il valore massimo tra i numeri acquisiti

► Inizializzazione di TOT

- Qual è il valore del massimo in un insieme di 0 numeri?
 - Non esiste, non è definito!
 - TOT = numero molto piccolo, che non possa certamente essere scambiato con il massimo

► Aggiornamento di TOT

- Sapendo che TOT è pari al massimo dei primi $i-1$ dati, e sapendo che il prossimo dato da considerare è d , quanto dovrà valere TOT ?
 - Se $d \leq TOT$, allora TOT rimane il massimo
 - Se $d > TOT$, allora il nuovo massimo sarà $TOT = d$

Esempio: massimo

```
int max ;

i = 0 ;
max = INT_MIN ;
while( i < N )
{
    scanf("%d", &dato) ;
    if(dato > max)
        max = dato ;

    i = i + 1 ;
}

printf("Massimo = %d\n", max);
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
                           delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```

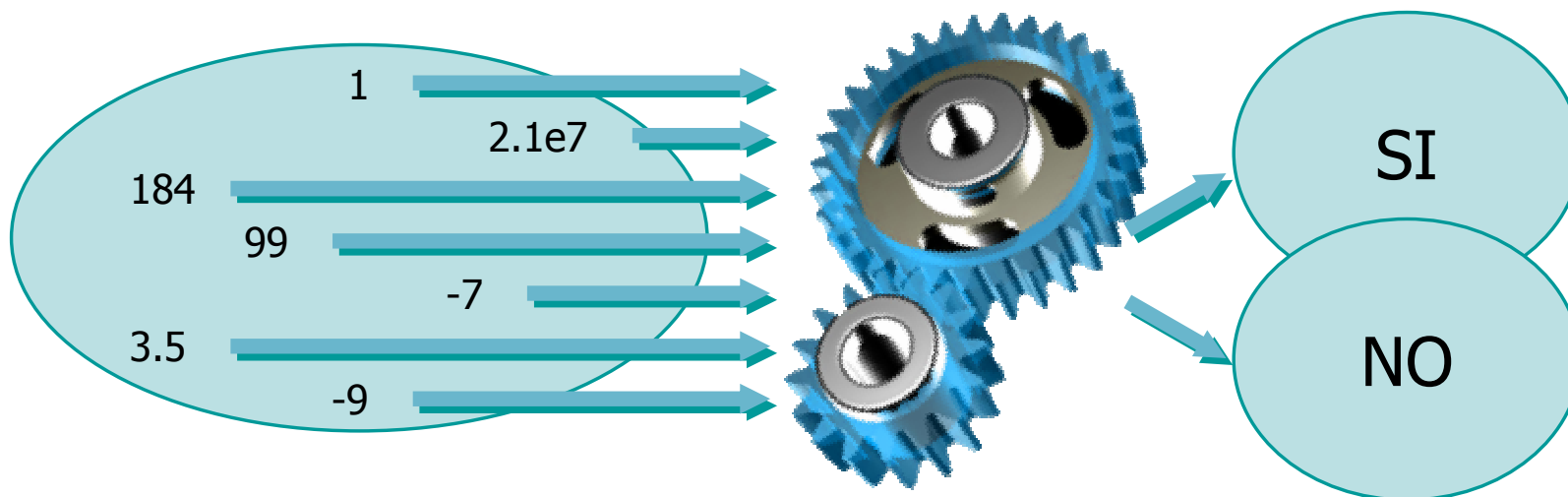


Schemi ricorrenti nei cicli

Flag

Flag, indicatori, variabili logiche

- Spesso occorre analizzare una serie di dati per determinare se si verifica una certa condizione
- Esempi:
 - Tra i dati inseriti esiste il numero 100?
 - Esistono due numeri consecutivi uguali?



Problemi

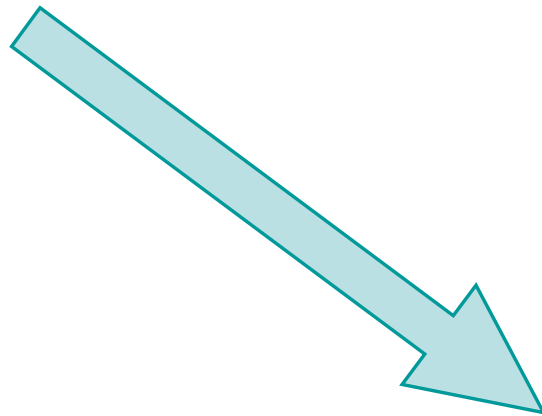
- Nel momento in cui si “scopre” il fatto, non si può interrompere l’elaborazione ma occorre comunque terminare il ciclo
- Al termine del ciclo, come fare a “ricordarsi” se si era “scoperto” il fatto o no?

Una possibile soluzione

- Per sapere
 - Se una certa condizione si verifica
è possibile contare
 - Quante volte quella condizione si verifica
ed in seguito verificare
 - Verificare se il conteggio è diverso da zero
- Ci riconduciamo ad un problema risolubile per mezzo di una variabile contatore

Esempio 1

Tra i dati inseriti esiste
il numero 100?



Conta quante volte tra i
dati inseriti compare il
numero 100

Il conteggio è > 0 ?

Soluzione (1/3)



trova100v1.c

```
int i, n ;
int dato ;
int conta ;
/* conta il numero di "100" letti */

printf("TROVA 100\n");

n = 10 ;

printf("Inserisci %d numeri\n", n);
```

Soluzione (2/3)

```
conta = 0 ;
```

```
i = 0 ;
```

```
while( i < n )  
{
```

```
    printf("Inserisci dato %d: ", i+1);  
    scanf("%d", &dato);
```

```
    if( dato == 100 )  
        conta = conta + 1 ;
```

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



trova100v1.c

Soluzione (3/3)

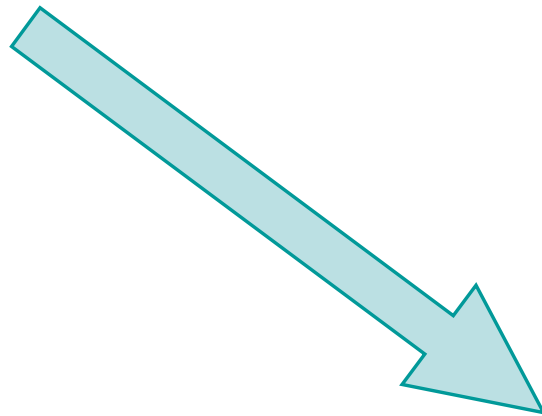


trova100v1.c

```
if( conta != 0 )  
    printf("Ho trovato il 100\n");  
else  
    printf("NON ho trovato il 100\n");
```

Esempio 2

Esistono due numeri
consecutivi uguali?



Conta quante volte due
numeri consecutivi sono
uguali

Il conteggio è > 0 ?

Soluzione (1/3)



ugualiv1.c

```
int i, n ;
int dato ;
int precedente ;
int conta ;
/* conta il numero di "doppioni" trovati */

printf("TROVA UGUALI\n");

n = 10 ;

printf("Inserisci %d interi\n", n);
```

Soluzione (2/3)



ugualiv1.c

```
conta = 0 ;

precedente = INT_MAX ;
/* ipotesi: l'utente non lo inserira' mai */

i = 0 ;
while( i < n )
{
    printf("Inserisci dato %d: ", i+1);
    scanf("%d", &dato);

    if( dato == precedente )
        conta = conta + 1 ;

    precedente = dato ;

    i = i + 1 ;
}
```

Soluzione (3/3)



ugualiv1.c

```
if( conta != 0 )
    printf("Ho trovato dei numeri  
consecutivi uguali\n");
else
    printf("NON ho trovato dei numeri  
consecutivi uguali\n");
```

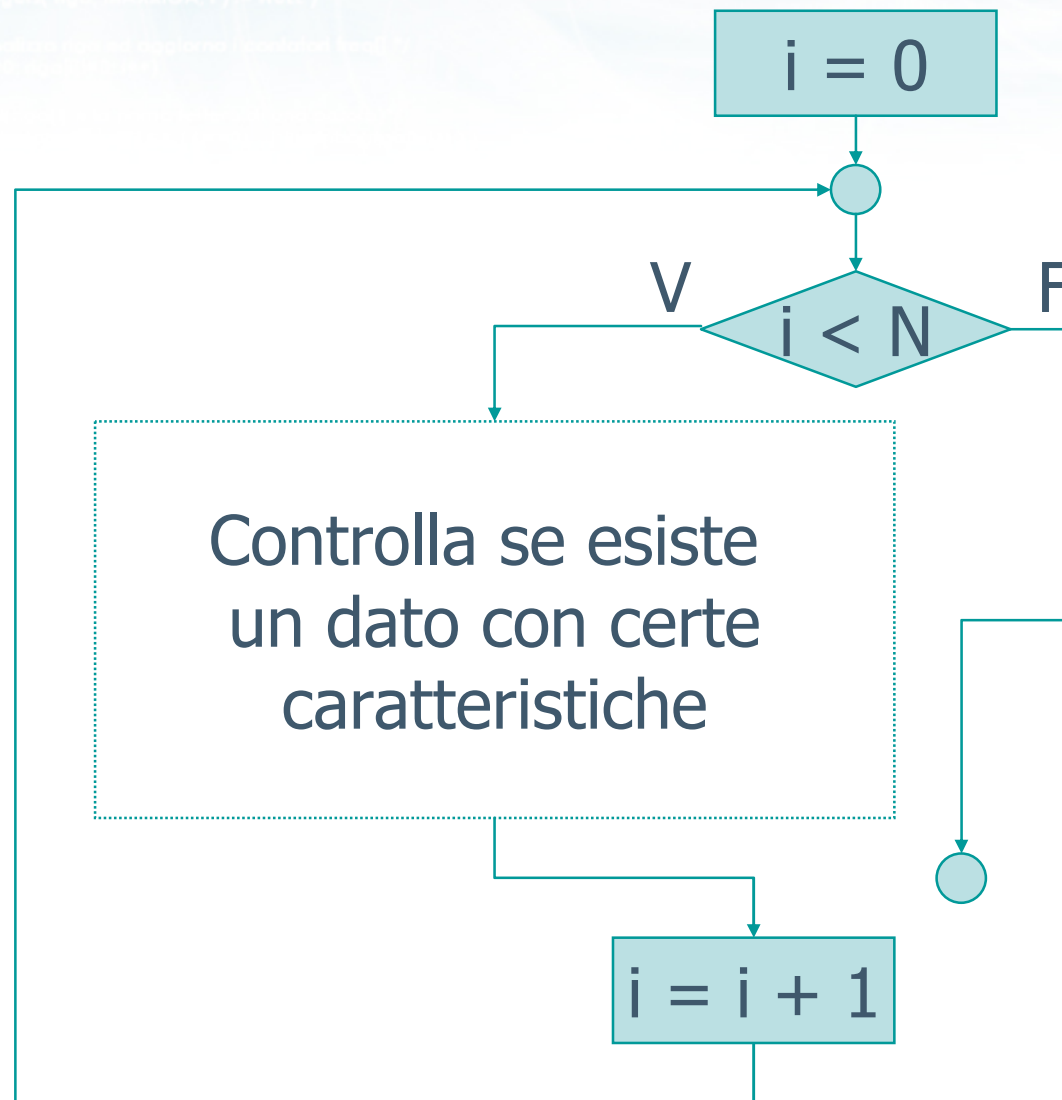
Svantaggi

- Il contatore determina quante volte si verifica la condizione ricercata
- In realtà non mi serve sapere quante volte, ma solo se si è verificata almeno una volta
- Usiamo un contatore “degenero”, che una volta arrivato ad 1 non si incrementa più

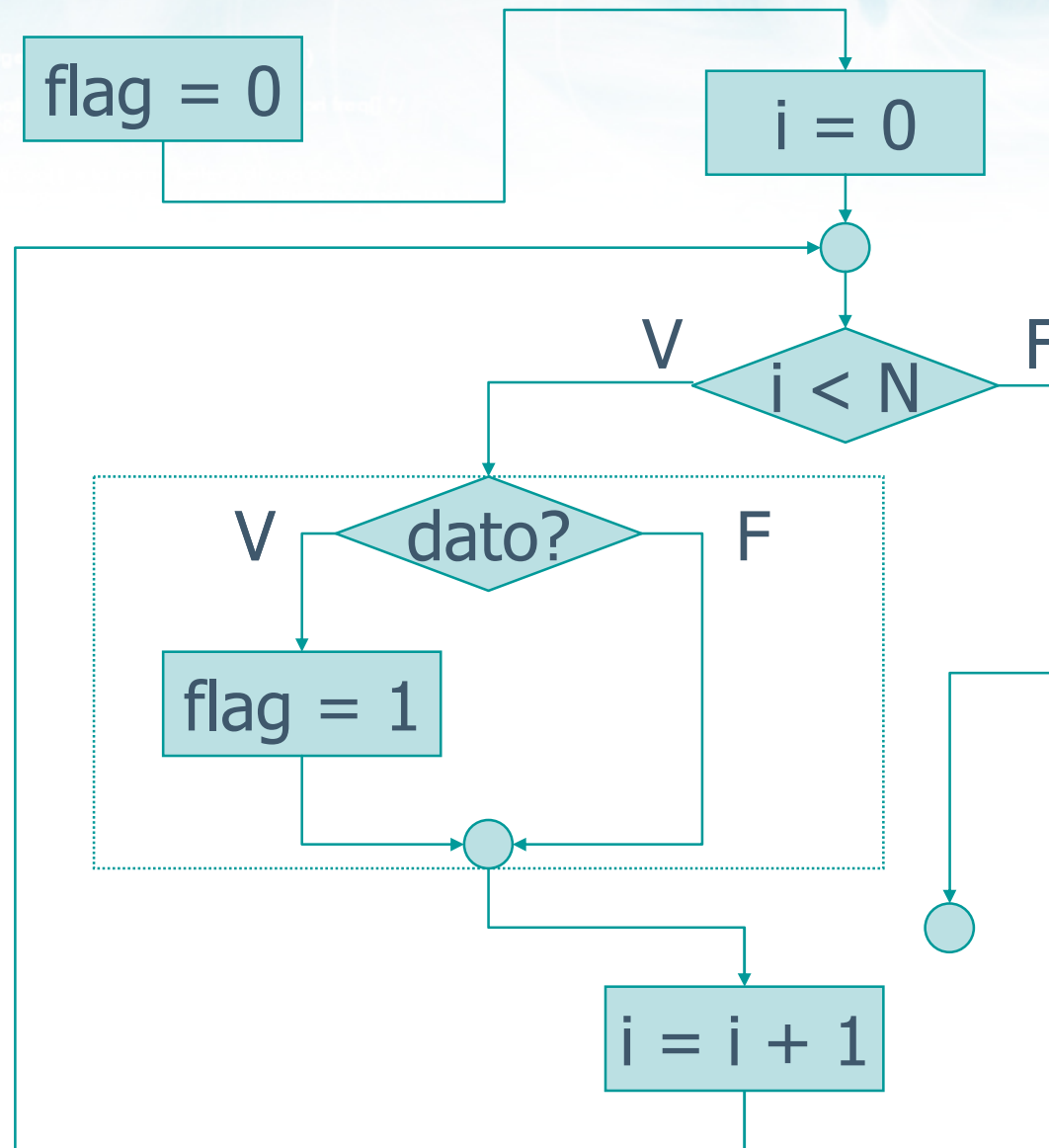


Variabili “flag”

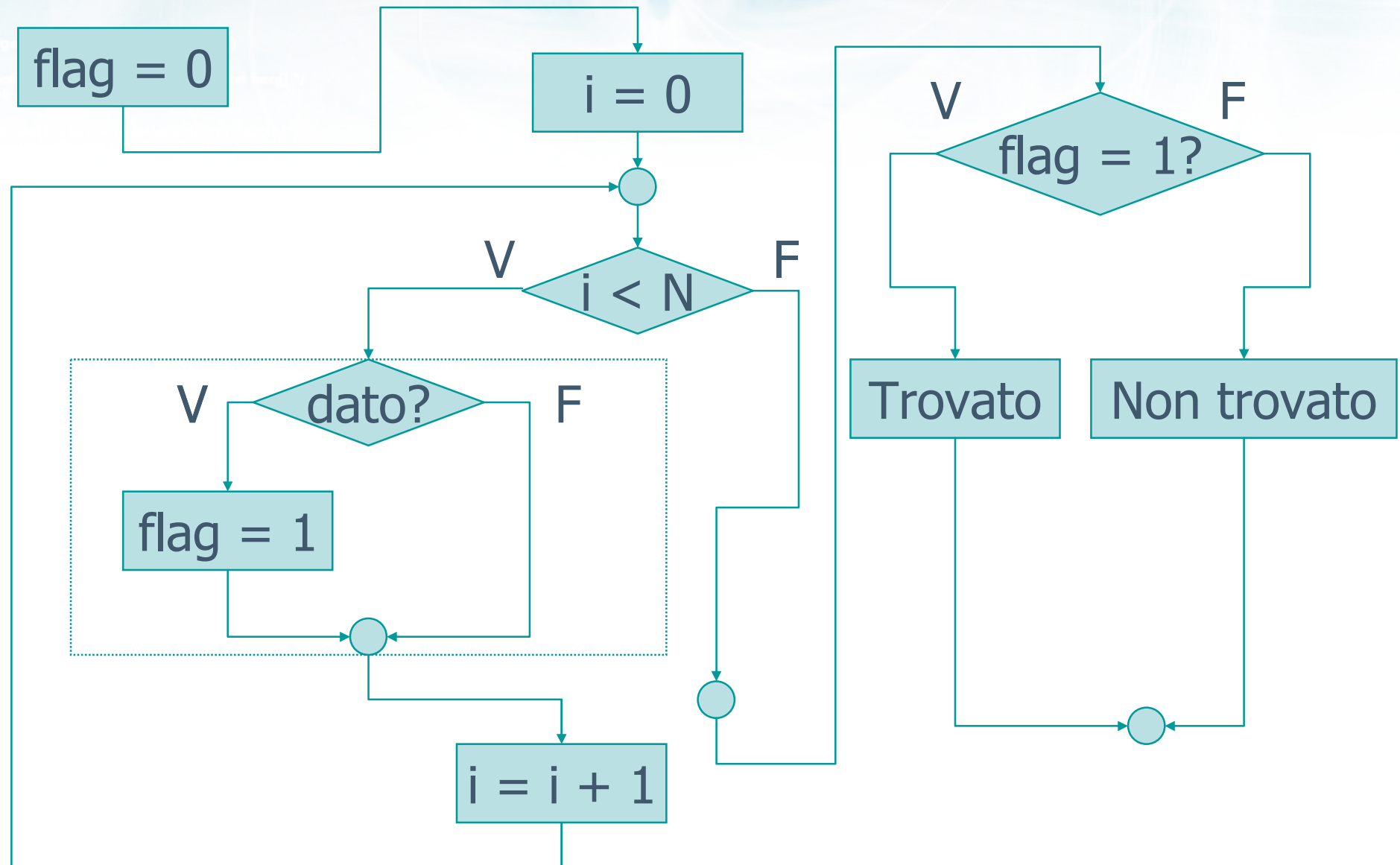
- Variabili intere che possono assumere solo due valori
 - Variabile = 0 $\Rightarrow$ la condizione non si è verificata
 - Variabile = 1 $\Rightarrow$ la condizione si è verificata
- Viene inizializzata a 0 prima del ciclo
- Se la condizione si verifica all'interno del ciclo, viene posta a 1
- Al termine del ciclo si verifica il valore
- Sinonimi: Flag, Variabile logica, Variabile booleana, Indicatore



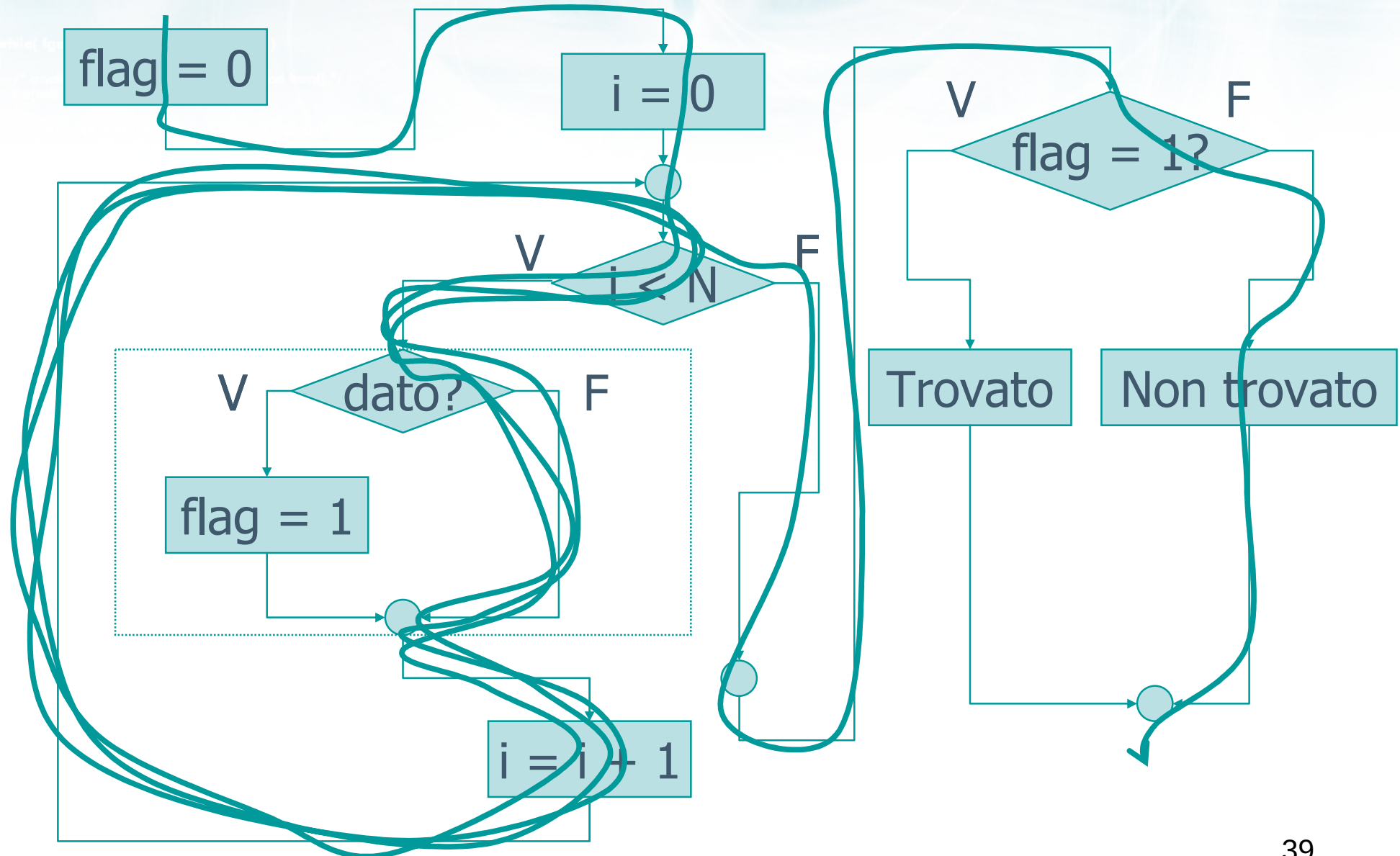
Analisi



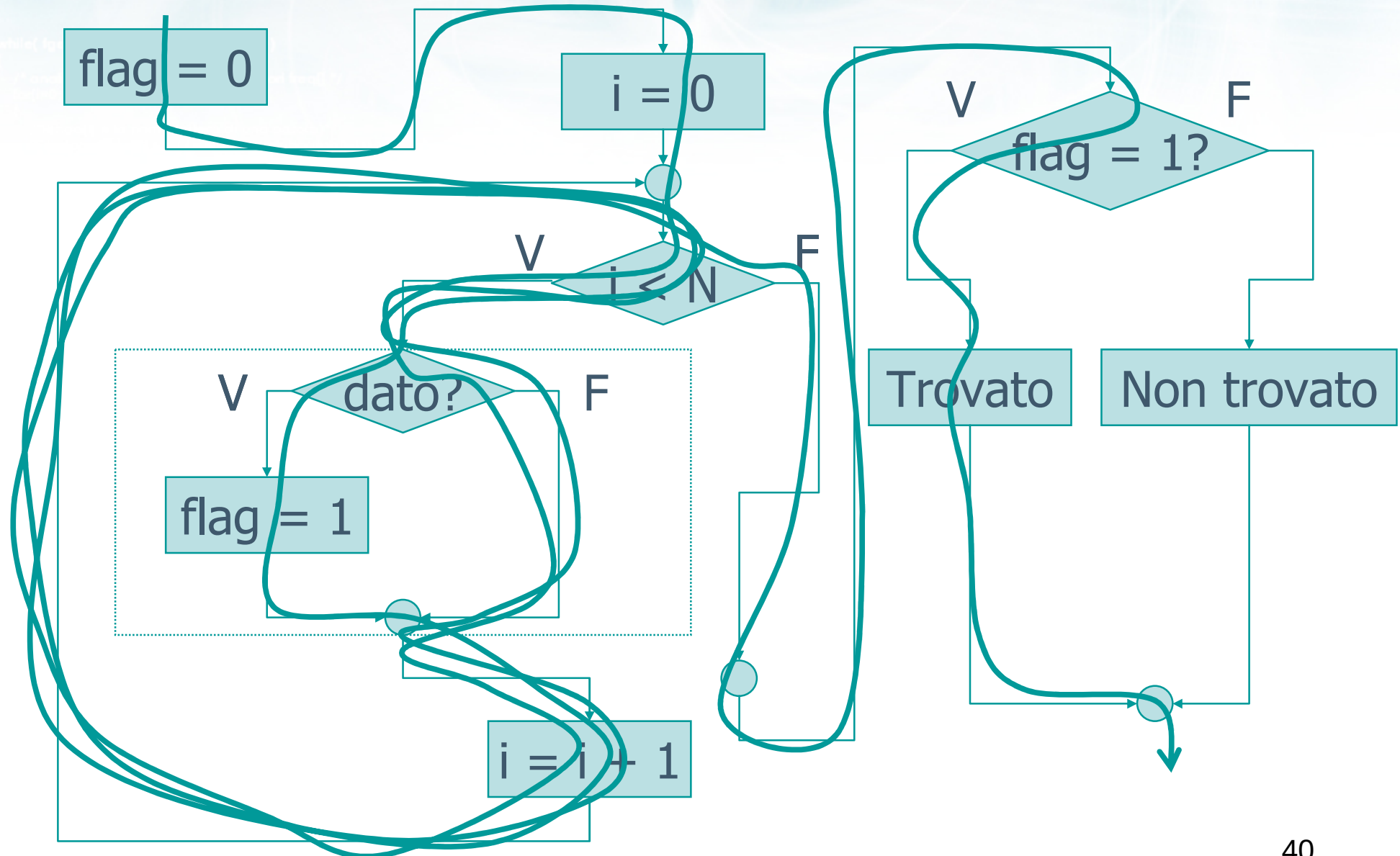
Analisi



Analisi



Analisi



Soluzione con flag – esempio 1

```
int trovato ; /* ho visto il numero "100"? */
....
trovato = 0 ;

i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == 100 )
        trovato = 1 ;

    i = i + 1 ;
}

if( trovato != 0 )
    printf("Trovato il numero 100\n");
else
    printf("NON trovato il numero 100\n");
```



trova100v2.c

Soluzione con flag – esempio 2

```
int doppi ; /* trovati "doppioni" ? */
...
doppi = 0 ;

precedente = INT_MAX ;
i = 0 ;
while( i < n )
{
    scanf("%d", &dato);

    if( dato == precedente )
        doppi = 1 ;

    precedente = dato ;
    i = i + 1 ;
}

if( doppi != 0 )
    printf("Trovati consecutivi uguali\n");
```



ugualiv2.c

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Schemi ricorrenti nei cicli

Esistenza e universalità

Ricerca di esistenza o universalità

- L'utilizzo dei flag è può essere utile quando si desiderino verificare delle proprietà su un certo insieme di dati
 - È vero che tutti i dati verificano la proprietà?
 - È vero che almeno un dato verifica la proprietà?
 - È vero che nessun dato verifica la proprietà?
 - È vero che almeno un dato non verifica la proprietà?

- Verificare che tutti i dati inseriti dall'utente siano positivi
- Determinare se una sequenza di dati inseriti dall'utente è crescente
- Due numeri non sono primi tra loro se hanno almeno un divisore comune
 - esiste almeno un numero che sia divisore dei due numeri dati
- Un poligono regolare ha tutti i lati di lunghezza uguale
 - ogni coppia di lati consecutivi ha uguale lunghezza

Formalizzazione

- È vero che tutti i dati verificano la proprietà?
 - $\forall x : P(x)$
- È vero che almeno un dato verifica la proprietà?
 - $\exists x : P(x)$
- È vero che nessun dato verifica la proprietà?
 - $\forall x : \text{not } P(x)$
- È vero che almeno un dato non verifica la proprietà?
 - $\exists x : \text{not } P(x)$

Realizzazione (1/2)

► Esistenza: $\exists x : P(x)$

- Inizializzo flag $F = 0$
- Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 1$
- Se $F = 1$, l'esistenza è dimostrata
- Se $F = 0$, l'esistenza è negata

Realizzazione (1/2)

➤ Esistenza: $\exists x : P(x)$

- Inizializzo flag $F = 0$

- Ciclo su tutte le x

- Se $P(x)$ è vera
 - Pongo $F = 1$

- Se $F = 1$, l'esistenza è dimostrata
- Se $F = 0$, l'esistenza è negata

➤ Universalità: $\forall x : P(x)$

- Inizializzo flag $F = 1$

- Ciclo su tutte le x

- Se $P(x)$ è falsa
 - Pongo $F = 0$

- Se $F = 1$, l'universalità è dimostrata
- Se $F = 0$, l'universalità è negata

Realizzazione (2/2)

- **Esistenza: $\exists x : \text{not } P(x)$**
 - Inizializzo flag $F = 0$
 - Ciclo su tutte le x
 - Se $P(x)$ è falsa
 - Pongo $F = 1$
 - Se $F = 1$, l'esistenza è dimostrata
 - Se $F = 0$, l'esistenza è negata
- **Universalità: $\forall x : \text{not } P(x)$**
 - Inizializzo flag $F = 1$
 - Ciclo su tutte le x
 - Se $P(x)$ è vera
 - Pongo $F = 0$
 - Se $F = 1$, l'universalità è dimostrata
 - Se $F = 0$, l'universalità è negata

Esempio 1

- Verificare che tutti i dati inseriti dall'utente siano positivi

```
int positivi ;  
...  
positivi = 1 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato <= 0 )  
        positivi = 0 ;  
    ....  
    i = i + 1 ;  
}  
if( positivi == 1 )  
    printf("Tutti positivi\n");
```

Esempio 2

- Determinare se una sequenza di dati inseriti dall'utente è crescente

```
int crescente ;  
...  
crescente = 1 ;  
precedente = INT_MIN ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato < precedente )  
        crescente = 0 ;  
    precedente = dato ;  
    ...  
    i = i + 1 ;  
}
```

Esempio 3

- Due numeri non sono primi tra loro se hanno almeno un divisore comune

```
int A, B ;
int noprimi ;
...
noprimi = 0 ;
i = 2 ;
while( i<=A )
{
    ...
    if( (A%i==0) && (B%i==0) )
        noprimi = 1 ;
    ....
    i = i + 1 ;
}
```


Esempio 4

- Un poligono regolare ha tutti i lati di lunghezza uguale

```
int rego ;
...
rego = 1 ;
precedente = INT_MIN ;
i = 0 ;
while( i < n )
{
    ...
    if( lato != precedente )
        rego = 0 ;
    precedente = lato ;
    ....
    i = i + 1 ;
}
```



Errore frequente

- Resettare il flag al valore di inizializzazione, dimenticando di fatto eventuali condizioni incontrate in precedenza

```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    else  
        trovato = 0 ;  
    ...  
    i = i + 1 ;  
}
```



```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
}
```



Errore frequente

- Passare ai fatti non appena trovato il primo elemento che soddisfa la proprietà

```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
    {  
        trovato = 1 ;  
        printf("w!\n");  
    }  
    ...  
    i = i + 1 ;  
}
```



```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
}  
if(trovato==1)  
    printf("w!\n");
```



Errore frequente

- Pensare che al primo fallimento si possa determinare che la proprietà è falsa

```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    else  
        printf("NO!\n");  
    ...  
    i = i + 1 ;  
}
```



```
trovato = 0 ;  
i = 0 ;  
while( i < n )  
{  
    ...  
    if( dato == 100 )  
        trovato = 1 ;  
    ...  
    i = i + 1 ;  
}  
if(trovato==0)  
    printf("NO!\n");
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Istruzione for

Istruzione for

- Sintassi dell'istruzione
- Operatori di autoincremento
- Cicli for annidati

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Istruzione for

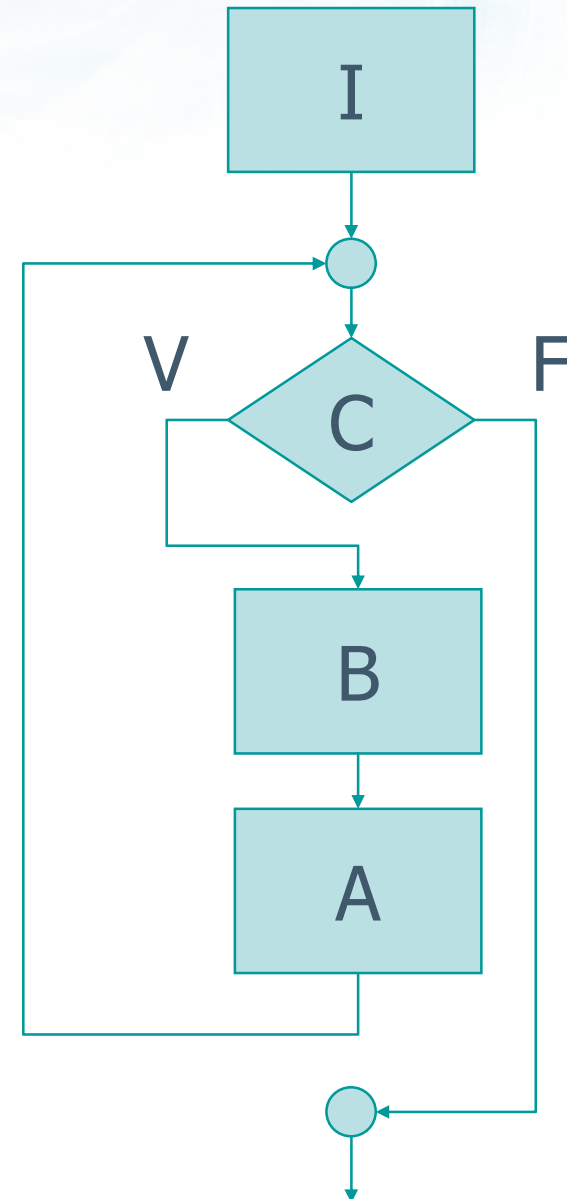
Sintassi dell'istruzione

Istruzione for

- L'istruzione fondamentale è `while`
 - La condizione solitamente valuta una variabile di controllo
 - Occorre ricordarsi l'inizializzazione della variabile di controllo
 - Occorre ricordarsi di aggiornare (incrementare, ...) la variabile di controllo
- L'istruzione `for` rende più semplice ricordare queste cose

Istruzione for

```
for ( I; C; A )  
{  
    B ;  
}
```



Istruzione for

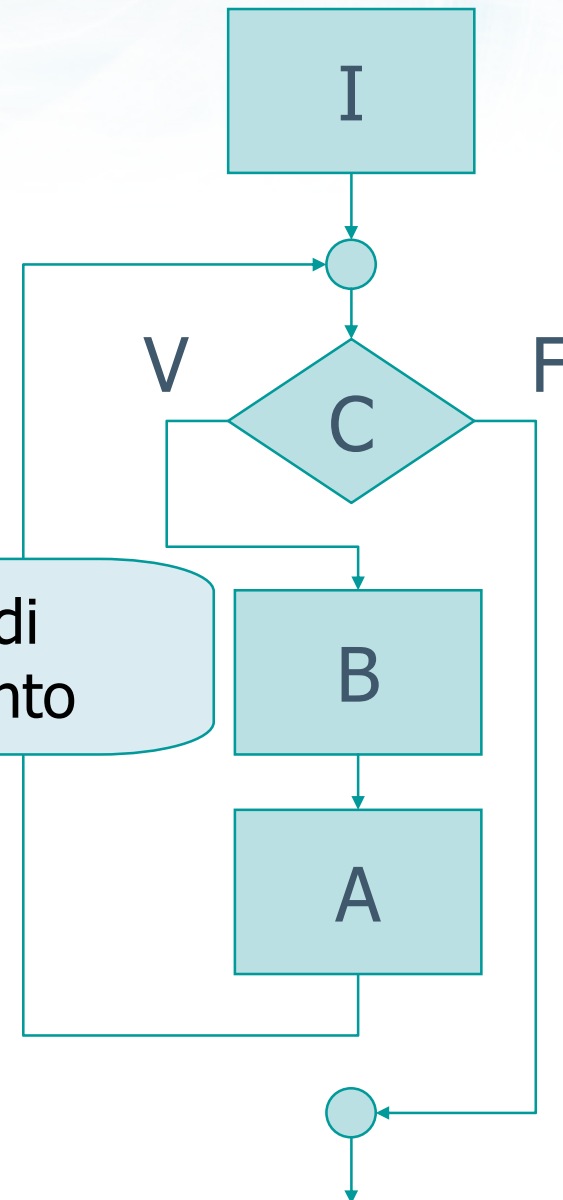
Istruzione di
inizializzazione

```
for ( I; C; A )  
{  
    B ;  
}
```

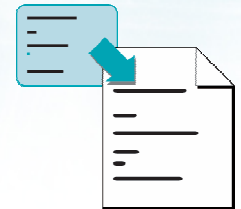
Istruzione di
aggiornamento

Corpo

Condizione



Esempio



num1-10v2.c

```
int i ;

for ( i=1; i<=10; i=i+1 )
{
    printf("Numero = %d\n", i) ;
}
```

Equivalenza $\text{for} \leftrightarrow \text{while}$

```
for ( I; C; A )  
{  
    B ;  
}
```

```
I ;  
while ( C )  
{  
    B ;  
    A ;  
}
```

Esempio

```
int i ;
```

```
for ( i=1; i<=10; i=i+1 )  
{  
    printf("%d\n", i) ;  
}
```

```
int i ;
```

```
i = 1 ;  
while ( i <= 10 )  
{  
    printf("%d\n", i) ;  
    i = i + 1 ;  
}
```

Utilizzo prevalente (1/2)

- Le istruzioni di inizializzazione **I** e di aggiornamento **A** possono essere qualsiasi
- Solitamente **I** viene utilizzata per inizializzare il contatore di controllo del ciclo, e quindi è del tipo
 - $i = 0$
- Solitamente **A** viene utilizzata per incrementare (o decrementare) il contatore, e quindi è del tipo
 - $i = i + 1$

```
for ( I; C; A )  
{  
    B ;  
}
```

Utilizzo prevalente (2/2)

- L'istruzione `for` può sostituire un qualsiasi ciclo `while`
- Solitamente viene utilizzata, per maggior chiarezza, nei cicli con numero di iterazioni noto a priori

Cicli for con iterazioni note

```
int i ;  
  
for ( i=0; i<N; i=i+1 )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=1; i<=N; i=i+1 )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=N; i>0; i=i-1 )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=N-1; i>=0; i=i-1 )  
{  
    .....  
}
```


Casi particolari (1/6)

- Se non è necessario inizializzare nulla, si può omettere l'istruzione **I**
 - `for( ; i != 0 ; i = i - 1 )`
 - La condizione **C** viene comunque valutata prima della prima iterazione, pertanto le variabili coinvolte dovranno essere inizializzate prima dell'inizio del ciclo
 - Il simbolo `;` è sempre necessario

```
for ( I; C; A )  
{  
    B ;  
}
```

Casi particolari (2/6)

- Se l'aggiornamento viene fatto nel ciclo, si può omettere l'istruzione **A**
 - `for( dato = INT_MIN; dato != 0 ; )`
 - La responsabilità di aggiornare la variabile di controllo (dato) è quindi del corpo **B** del ciclo
 - Il simbolo `;` è sempre necessario

```
for ( I; C; A )  
{  
    B ;  
}
```

Casi particolari (3/6)

- Se occorre inizializzare più di una variabile, è possibile farlo separando le varie inizializzazioni con un simbolo ,
 - `for( i=0, j=0; i<N; i=i+1 )`
 - Solitamente uno solo è il contatore del ciclo, gli altri saranno altri contatori, accumulatori o flag

```
for ( I; C; A )  
{  
    B ;  
}
```

Casi particolari (4/6)

- Se occorre aggiornare più di una variabile, è possibile farlo separando i vari aggiornamenti con un simbolo ,
 - `for( i=0; i<N; i=i+1, k=k-1 )`

```
for ( I; C; A )  
{  
    B ;  
}
```

Casi particolari (5/6)

- Nel caso in cui si ometta sia **I** che **A**, il ciclo **for** degenera nel ciclo **while** e equivalente
 - **for(; i<N;)**
 - **while(i<N)**

```
for ( I; C; A )  
{  
    B ;  
}
```

Casi particolari (6/6)

- È possibile omettere la condizione **C**, in tal caso viene considerata come sempre vera
 - `for( i=0; ; i=i+1)`
 - Questo costrutto genera un ciclo infinito. È necessario che il ciclo venga interrotto con un altro meccanismo (`break`, `return`, `exit`)
 - Talvolta si incontra anche un ciclo infinito “puro”
 - `for( ; ; )`

```
for ( I; C; A )  
{  
    B ;  
}
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Istruzione for

Operatori di autoincremento

Istruzione di aggiornamento

- Nella maggioranza dei casi, l'istruzione di aggiornamento **A** consiste in un incremento
 - $i = i + 1$
- oppure in un decremento
 - $i = i - 1$
- Il linguaggio **C** dispone di operatori specifici per semplificare la sintassi di queste operazioni frequenti

```
for ( I; C; A )  
{  
    B ;  
}
```


Operatore di auto-incremento

`a++ ;`

`++a ;`

`a = a + 1 ;`

`a-- ;`

`--a ;`

`a = a - 1 ;`

Cicli for con iterazioni note

```
int i ;  
  
for ( i=0; i<N; i++ )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=1; i<=N; i++ )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=N; i>0; i-- )  
{  
    .....  
}
```

```
int i ;  
  
for ( i=N-1; i>=0; i-- )  
{  
    .....  
}
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



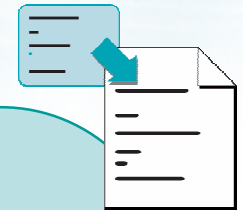
Istruzione for

Cicli for annidati

Annidamento di cicli

- Come sempre, all'interno del corpo **B** di un ciclo (for o while) è possibile annidare altri cicli (for o while)
- Non vi è limite al livello di annidamento
- I cicli più interni sono sempre eseguiti "più velocemente" dei cicli più esterni

Esempio



conta99v2.c

```
for( i=0; i<N; i++ )
{
    for( j=0; j<N; j++ )
    {
        printf("i=%d - j=%d\n", i, j);
    }
}
```

- Si scriva un programma in linguaggio C che
 - acquisisca da tastiera 10 numeri
 - per ciascuno di tali numeri determini se è un numero primo, stampando immediatamente un messaggio opportuno
 - al termine, se nessuno tra i numeri inseriti era un numero primo, stampi un messaggio opportuno

Analisi (1/2)

C:\> Prompt dei comandi

TROVA PRIMI

Inserisci 4 numeri interi

Inserisci dato 1: 6

Inserisci dato 2: 3

E' un numero primo

Inserisci dato 3: 4

Inserisci dato 4: 5

E' un numero primo

Analisi (2/2)

```
C:\> Prompt dei comandi

TROVA PRIMI
Inserisci 4 numeri interi

Inserisci dato 1: 4
Inserisci dato 2: 6
Inserisci dato 3: 8
Inserisci dato 4: 9

Non c'erano numeri primi
```


Numero primo



10primi.c

```
primo = 1 ;  
for( j=2; j<dato; j++)  
{  
    if( dato%j == 0 )  
        primo = 0 ;  
}
```

Stampa se non ci sono primi

```
for( i=0; i<n; i++ )  
{  
    scanf("%d", &dato);
```

....determina se è un numero primo....

```
if( primo == 1 )  
{  
    printf("E' un numero primo\n");  
}  
}
```



10primi.c

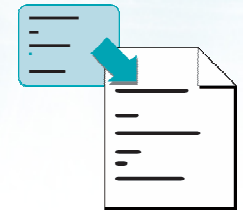
Stampa se è un primo



10primi.c

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    ....acquisisci dato e determina se è un numero primo....
    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}
if( trovato == 0 )
    printf("Non ci sono primi\n") ;
```

Vista d'insieme



10primi.c

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
    printf("Non c'erano numeri primi\n");
```

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
    printf("Non c'erano numeri primi\n");
```

Ciclo esterno

Ciclo interno

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
    printf("Non c'erano numeri primi\n");
```

Flag interno:
numero primo

Inizializzazione

Aggiornamento

Verifica

Vista d'insieme

```
trovato = 0 ;
for( i=0; i<n; i++ )
{
    scanf("%d", &dato);

    primo = 1 ;
    for( j=2; j<dato; j++ )
    {
        if( dato%j == 0 )
            primo = 0 ;
    }

    if( primo == 1 )
    {
        printf("E' un numero primo\n");
        trovato = 1 ;
    }
}

if( trovato == 0 )
    printf("Non c'erano numeri primi\n");
```

Flag esterno:
nessun primo

Inizializzazione

Aggiornamento

Verifica

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Approfondimenti

Approfondimenti

- Istruzione `do-while`
- Istruzione `break`
- Istruzione `continue`

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```

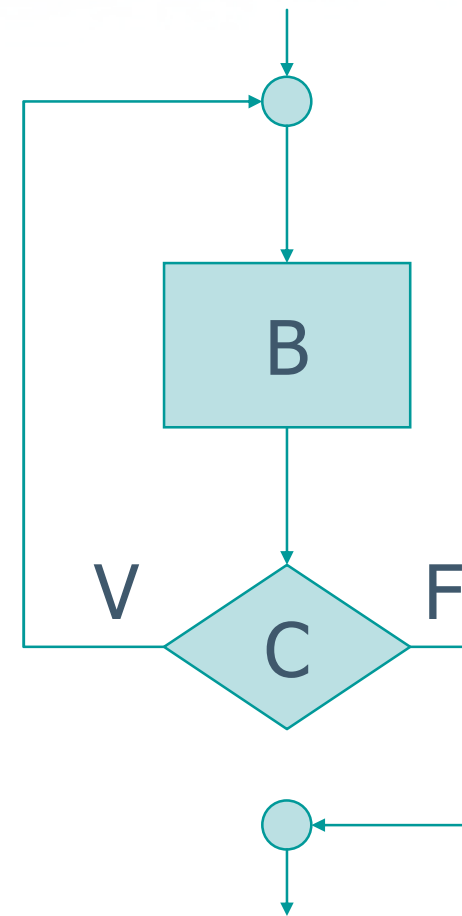


Approfondimenti

Istruzione do-while

Istruzione do-while

```
do {  
    B ;  
} while ( C ) ;
```



➤ Istruzione `while`

- Condizione valutata prima di ogni iterazione
- Numero minimo di iterazioni: 0
- Per uscire: condizione falsa

➤ Istruzione `do-while`

- Condizione valutata al termine di ogni iterazione
- Numero minimo di iterazioni: 1
- Per uscire: condizione falsa

Esempio

- Acquisire un numero compreso tra 1 e 10 da tastiera
- Nel caso in cui l'utente non inserisca il numero correttamente, chiederlo nuovamente

Soluzione

```
printf("Numero tra 1 e 10\n");  
do {  
  
    scanf("%d", &n) ;  
  
} while ( n<1 || n>10 ) ;
```

Soluzione migliore

```
printf("Numero tra 1 e 10\n");  
do {  
  
    scanf("%d", &n) ;  
  
    if( n<1 || n>10 )  
        printf("Errore: ripeti\n");  
  
} while ( n<1 || n>10 ) ;
```

Esempio

- Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- La sequenza termina quando l'utente inserisce il dato 9999

Soluzione

```
somma = 0 ;  
do {  
  
    scanf("%d", &dato) ;  
  
    if( n != 9999 )  
        somma = somma + dato ;  
  
} while ( dato != 9999 ) ;
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
```

```
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
```

```
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
```

```
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Approfondimenti

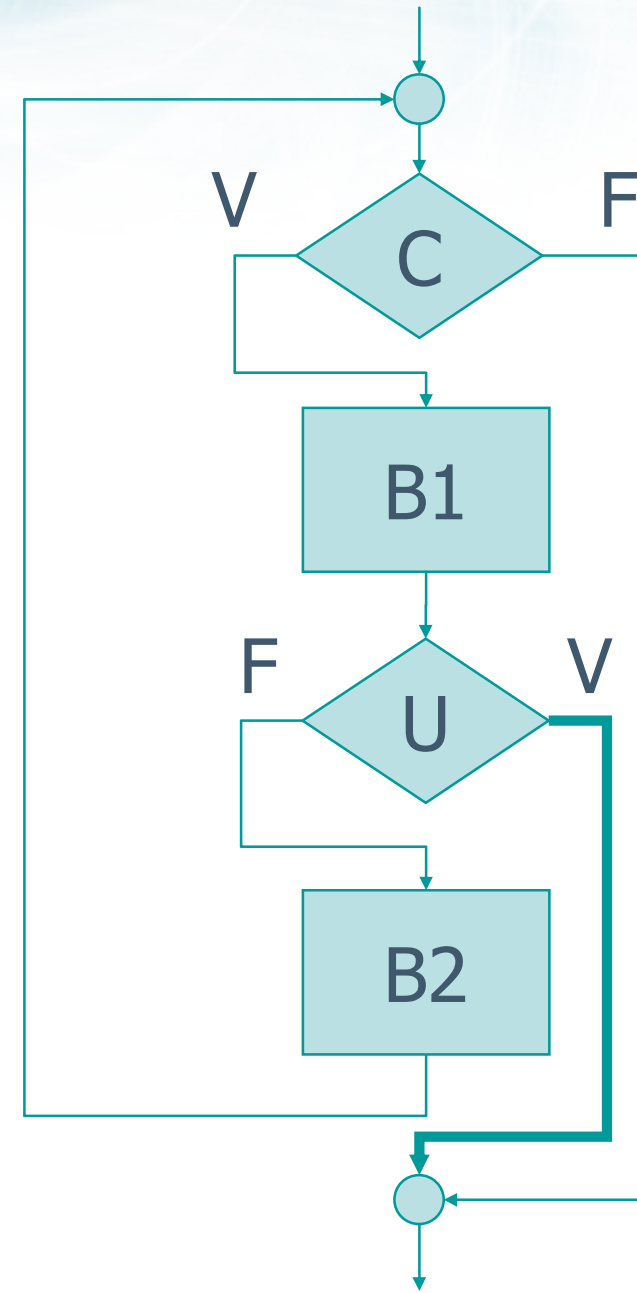
Istruzione break

Interruzione dei cicli

- Di norma, un ciclo termina quando la condizione di controllo diventa falsa
 - Necessario arrivare al termine del corpo per poter valutare la condizione
- Talvolta potrebbe essere comodo interrompere prematuramente l'esecuzione di un ciclo
 - A seguito di condizioni di errore
 - Quando è stato trovato ciò che si cercava

Istruzione break

```
while ( C )  
{  
    B1 ;  
    if ( U )  
        break ;  
    B2 ;  
}
```



- Quando viene eseguita l'istruzione `break`
 - Viene interrotta l'esecuzione del corpo del ciclo
 - Il flusso di esecuzione passa all'esterno del ciclo che contiene il `break`
 - Si esce dal ciclo anche se la condizione di controllo è ancora vera
 - In caso di cicli annidati, si esce solo dal ciclo più interno
- Funziona con cicli `while`, `for`, `do-while`

Esempio

- Si scriva un programma in C che calcoli la somma di una sequenza di numeri interi
- La sequenza termina quando l'utente inserisce il dato 9999

Soluzione

```
somma = 0 ;  
do {  
  
    scanf("%d", &dato) ;  
  
    if( dato == 9999 )  
        break;  
  
    somma = somma + dato ;  
  
} while ( 1 ) ;
```

Esempio

- Si scriva un programma in C che determini se un numero inserito da tastiera è primo

Soluzione

```
scanf("%d", &dato) ;
primo = 1 ;
for ( i=2; i<dato; i++ )
{
    if( dato%i == 0 )
    {
        primo = 0 ;
        break ; /* inutile continuare */
    }
}
```

- L'istruzione `break` crea programmi non strutturati: usare con cautela
- Non è possibile uscire da più cicli annidati contemporaneamente

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```

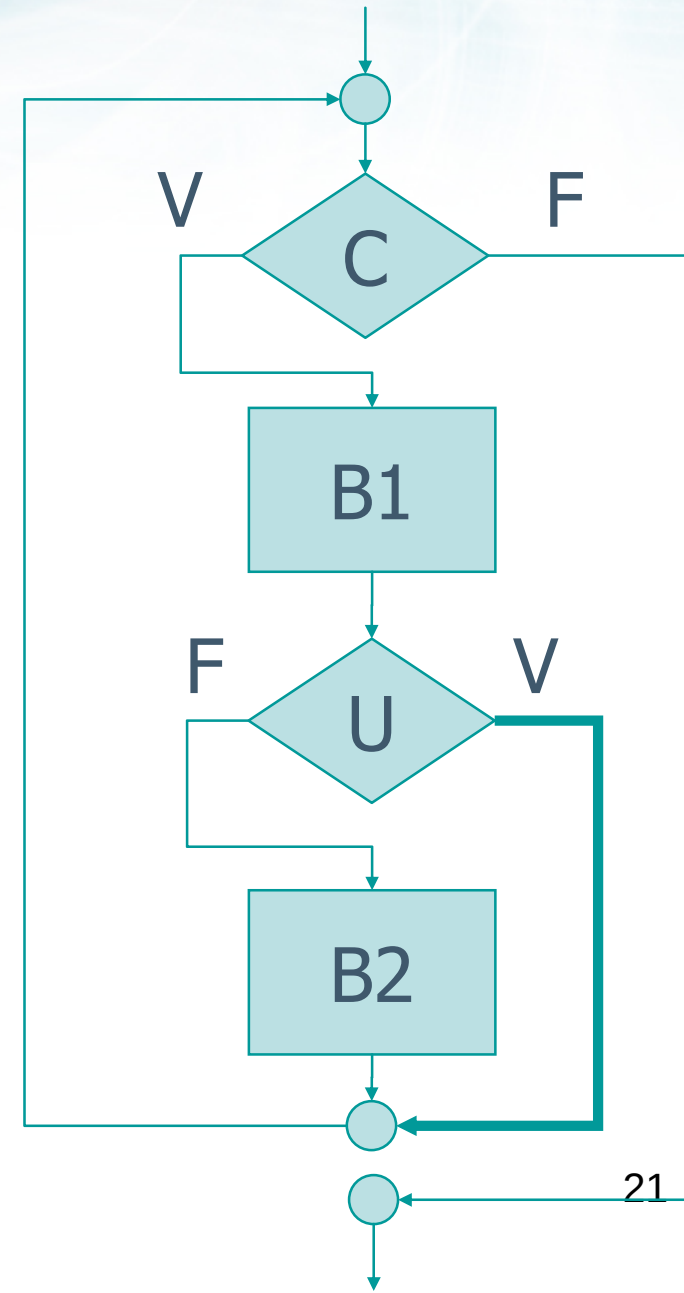


Approfondimenti

Istruzione continue

Istruzione continue

```
while ( C )  
{  
    B1 ;  
    if ( U )  
        continue ;  
    B2 ;  
}
```



- Quando viene eseguita l'istruzione `continue`
 - Viene interrotta l'esecuzione del corpo del ciclo
 - Il flusso di esecuzione passa al termine del corpo
 - Nel caso di cicli `for`, viene eseguita l'istruzione di aggiornamento
 - Viene nuovamente valutata la condizione
 - Il ciclo continua normalmente
 - In caso di cicli annidati, si esce solo dal ciclo più interno
- Funziona con cicli `while`, `for`, `do-while`

Esempio

```
somma = 0 ;  
do {  
  
    scanf("%d", &dato) ;  
  
    if( dato == 9999 )  
        continue ; /* non considerarlo */  
  
    somma = somma + dato ;  
  
} while ( dato != 9999 ) ;
```

Avvertenze

- L'istruzione `continue` è oggettivamente poco utilizzata
- Crea un "salto" poco visibile: accompagnarla sempre con commenti evidenti


```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Esercizi proposti

Esercizi proposti

- Esercizio "Decimale-binario"
- Esercizio "Massimo Comun Divisore"
- Esercizio "Triangolo di Floyd"

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Esercizi proposti

Esercizio “Decimale-binario”

Esercizio “Decimale-binario”

- Si realizzi un programma in C in grado di
 - Leggere un numero naturale n
 - Convertire tale numero dalla base 10 alla base 2
 - Visualizzare il risultato, a partire dalla cifra meno significativa

C:\> Prompt dei comandi

DECIMALE – BINARIO

Inserisci un numero intero positivo: 12

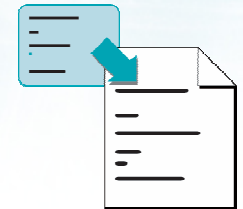
Numero binario: 0 0 1 1

Divisioni successive

- $n = 12$
- $n \% 2 = 0 \rightarrow$ cifra 0
- $n = n / 2 = 6$
- $n \% 2 = 0 \rightarrow$ cifra 0
- $n = n / 2 = 3$
- $n \% 2 = 1 \rightarrow$ cifra 1
- $n = n / 2 = 1$
- $n \% 2 = 1 \rightarrow$ cifra 1
- $n = n / 2 = 0 \rightarrow$ STOP

N	N % 2
12	0
6	0
3	1
1	1
0	

Soluzione



bin-dec.c

```
while( n!=0 )
{
    if( n%2 == 1 )
        printf("1 ") ;
    else
        printf("0 ") ;

    n = n / 2 ;
}
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Esercizi proposti

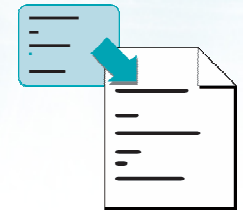
Esercizio “Massimo Comun Divisore”

Esercizio “Massimo Comun Divisore”

- Si scriva un programma in C in grado di calcolare il massimo comun divisore (MCD) di due numeri interi.
- Il MCD è definito come il massimo tra i divisori comuni ai due numeri.

- Diciamo N1 e N2 i numeri inseriti dall'utente
- Il MCD di N1 e N2 è il **massimo** tra i numeri che sono **divisori** sia di N2, sia di N1.
 - Troviamo i divisori di N1 ...
 - ... tra quelli che sono anche divisori di N2 ...
 - ... calcoliamo il massimo

Algoritmo



bin-dec.c

- $k_max = 0$
- for $k =$ da 1 a $N1$
 - se k è un divisore di $N1$
 - se k è un divisore di $N2$
 - aggiorna $k_max = k$
- $MCD = k_max$

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Esercizi proposti

Esercizio “Triangolo di Floyd”

Esercizio "Triangolo di Floyd"

- Scrivere un programma C per la rappresentazione del triangolo di Floyd.
- Il programma riceve da tastiera un numero intero N.
- Il programma visualizza le prima N righe del triangolo di Floyd.

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

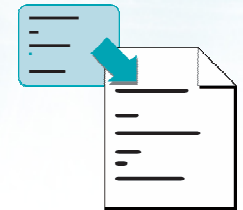
N=5

- Occorre stampare i primi numeri interi in forma di triangolo
- La riga k-esima ha k elementi

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

N=5

Algoritmo



floyd.c

- cont = 1
- for riga = da 1 a N
 - for colonna = da 1 a riga
 - stampa cont
 - cont++
 - vai a capo

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

N=5

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Cicli ed iterazioni

Sommario

Argomenti trattati

- Ripetizione del del flusso di esecuzione
- Inizializzazione, Condizione, Aggiornamento, Corpo
- Istruzione `while`
- Istruzione `for`
- Cicli annidati

Tecniche di programmazione

- Cicli con numero di iterazioni note o ignote
- Contatori
- Accumulatori
- Flag

Schemi ricorrenti

- Calcolo di somme, medie, ...
- Calcolo di max, min
- Ricerca di esistenza
- Ricerca di universalità
- Controllo dei dati in input



Suggerimenti

- Ricordare di verificare sempre le 4 parti del ciclo
 - Inizializzazione, Condizione, Corpo, Aggiornamento
- Le complicazioni nascono da
 - Cicli annidati
 - Condizioni if annidate in cicli
 - Annidamento di flag o ricerche
- Procedere sempre per gradi
 - Pseudo-codice o flow chart
 - Identificare chiaramente il ruolo dei diversi cicli

Materiale aggiuntivo

➤ Sul CD-ROM

- Testi e soluzioni degli esercizi trattati nei lucidi
- Scheda sintetica
- Esercizi risolti
- Esercizi proposti

➤ Esercizi proposti da altri libri di testo