3. PSEUDOCODIFICA DI UN ALGORITMO

In informatica per descrivere un algoritmo viene utilizzato:

a) un linguaggio, basato sull'utilizzo di simboli grafici, per descrivere i singoli passi dell'algoritmo, detto **linguaggio dei diagrammi a blocchi o flow chart** (oppure **diagramma di flusso**);
b) un linguaggio, molto vicino al linguaggio naturale, detto **linguaggio di progetto o pseudolinguaggio** talvolta abbastanza vicino ad alcuni linguaggi di programmazione.

La descrizione dell'algoritmo effettuata con lo pseudolinguaggio prende il nome di **pseudocodifica**.

Tale attività è una fase intermedia che si frappone tra la fase di analisi del problema e quella di codifica in un vero e proprio linguaggio di programmazione. Lo scopo principale della pseudocodifica è di portare l'utente ad esprimere le proprie istruzioni in una forma naturale, utilizzando frasi ed espressioni elementari della lingua italiana. Ciò permette di concentrarsi sulla soluzione logica del problema invece che sulla forma e sui vincoli da rispettare nella sua enunciazione.

Valgono le seguenti regole generali:

1) le **parole chiave** o **riservate** saranno scritte in maiuscolo;
2) gli **identificatori** (ossia i nomi) delle **variabili** saranno scritti sempre in *minuscolo* e quelli delle costanti sempre in *maiuscolo*;
3) le parole racchiuse tra **parentesi angolari** < > rappresentano le categorie sintattiche ossia elementi generali del linguaggio che nei vari algoritmi saranno sostituiti con opportune occorrenze;
4) i blocchi racchiusi tra **parentesi quadre [ ]** indicano l'opzionalità ossia tali blocchi possono anche non essere presenti;
5) i blocchi separati dal simbolo | possono essere usati in alternativa (esclusiva).

Lo pseudocodice di per sé non può essere caricato direttamente in un calcolatore per essere eseguito ma dovrà essere tradotto in un codice scritto in un linguaggio (linguaggio di programmazione) che possa essere capito dal computer (esempio C, PASCAL, etc.).
I vantaggi connessi con l'attività di pseudocodifica sono grandissimi tra i quali segnaliamo:
- la possibilità di **comunicare la logica** dell'algoritmo a persone che non sono dei tecnici;
- la possibilità di **esprimere la logica** senza le limitazioni ed i vincoli tipici dei linguaggi di programmazione;
- la possibilità di **usare lo stesso pseudocodice** per passare poi a diversi linguaggi di programmazione.

Spesso le istruzioni dello pseudocodice vengono indicate con il termine di **pseudoistruzioni**.

SIMBOLOGIA PER LA PSEUDOCODIFICA DI UN ALGORITMO

TIPI DI DATO SEMPLICE

INT	per numerico intero
REAL	per numerico reale
CHAR	per singolo carattere alfanumerico
BOOL	per valore logico (VERO o FALSO)

TIPI DI DATO STRUTTURATO

ARRAY MONO e BIDIMENSIONALE
RECORD
MATRICE
TABELLA

OPERATORI ARITMETICI

+	per l'addizione
-	per la sottrazione
*****	per la moltiplicazione
/	per la divisione
%	per il resto (modulo n)

OPERATORI RELAZIONALI

=	per "*uguale a*"
<	per "*minore di*
<=	pr "*minore o uguale a*"
>	per "*maggiore di*"
>=	per "*maggiore o uguale a*"
!=	per "*diverso da*"

OPERATORI LOGICI

AND	per l'and logico
OR	per l'or inclusivo
NOT	per la negazione
XOR	per l'or esclusivo

COMMENTO

/* *<qui si scrive la riga di commento>* ***/** **oppure**
// *<qui si scrive la riga di commento>*

ISTRUZIONE DI ASSEGNAZIONE

<variabile > ← <variabile> | <costante> | <espressione>

ISTRUZIONI DI I/O (input/output)

Leggi (<variabile>) oppure **Scrivi** (<variabile> | <costante>)

STRUTTURE DI CONTROLLO

a) SEQUENZA

INIZIO
……
<istruzione-1>
<istruzione-2>
<istruzione-3>
……
FINE

b) SELEZIONE

b1) Selezione ad una scelta o **unaria**

SE (<condizione>)
ALLORA
<B1>
FINE SE

b2) Selezione a due scelte o **binaria**

SE (<condizione>)
ALLORA
<B1>
ALTRIMENTI
<B2>
FINE SE

b3) Selezione a più scelte o **ennaria**

NEL CASO CHE (<variabile> | <espressione>) **SIA**
<lista valori 1> **:** <B1>
<lista valori 2> **:** <B2>
…… : ……
<lista valori n> **:** <B_n>
[ALTRIMENTI : <B_{n+1}>]
FINE CASO

c) ITERAZIONE

c1) iterazione con test all'inizio sulla condizione per valore VERO

MENTRE (<condizione>) **ESEGUI**
<B1>
FINE MENTRE

c2) iterazione con test alla fine sulla condizione per valore FALSO

RIPETI
<B1>
FINCHE' (<condizione>)

c3) iterazione con numero di iterazioni noto a priori

PER <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI**
<B1>
INCREMENTA <indice> **oppure** <indice> ← <indice> **+** 1
[DECREMENTA <indice>**]** **oppure** <indice> ← <indice> **-** 1
FINE PER

ALGORITMO PRINCIPALE

ALGORITMO <Nome dell'algoritmo>

PROCEDURA main( )
<sezione dichiarativa oggetti locali alla procedura main ( ) >

INIZIO

<corpo dell'algoritmo>

[**RITORNA**]
FINE

SOTTOPROGRAMMI

1) PROCEDURE

PROCEDURA <Nome procedura> **[(REF | VAL** <Nome param 1>**:** <Tipo param 1> **,**
REF | VAL <Nome param 2**>:** <Tipo param 2> ,
……
REF | VAL <Nome param n**>:** <Tipo param n> **)]**

< sezione dichiarativa oggetti locali alla procedura>

INIZIO

< corpo della procedura>

RITORNA
FINE

2) FUNZIONI

FUNZIONE <Nome funzione> **[(REF | VAL** <Nome param 1>**:** <Tipo param 1> **,**
REF | VAL <Nome param 2**>:** <Tipo param 2> ,
……
REF | VAL <Nome param n**>:** <Tipo param n> **)]**
: < Tipo Risultato>

< sezione dichiarativa oggetti locali alla funzione>

INIZIO

< corpo della funzione>
RITORNA <risultato>

FINE

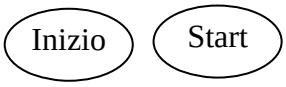
I FLOW CHART (o Diagrammi di Flusso o Diagrammi a Blocchi)

I flow chart sono schemi che descrivono visivamente come procede l'esecuzione di un programma. Essi non sono legati ad uno specifico linguaggio: dato un flow chart, il programmatore può poi usare un qualsiasi linguaggio di programmazione (si tratta, per così dire, di un linguaggio visuale comprensibile a tutti i programmatori). Il flow chart aiuta anche il programmatore a descrivere correttamente un algoritmo (il procedimento risolutivo di un problema).

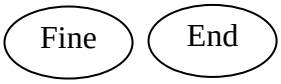
Ogni tipo di istruzione che si può inserire in un programma ha un suo simbolo ed ognuna delle tre strutture fondamentali della programmazione (sequenza, selezione ed iterazione) può essere rappresentata. Esistono anche simboli speciali (inizio programma, fine programma ecc.) che non rappresentano istruzioni vere e proprie ma che sono utili per la costruzione del flow chart.

I PRINCIPALI SIMBOLI

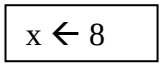
Inizio programma (i simboli sono equivalenti)



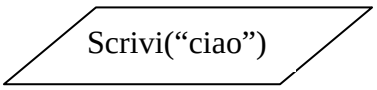
Fine programma (i simboli sono equivalenti)



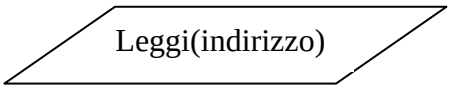
Assegnazione o altre istruzioni generiche
(esempio: dai ad x il valore 8)



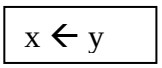
Scrittura di un valore sul video
(esempio: mostra a video la scritta "ciao")



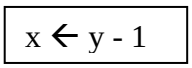
Lettura di un valore scritto con la tastiera e memorizzato nella variabile indicata
(esempio: leggere dove abita una persona e memorizzare l'informazione nella variabile *indirizzo)*



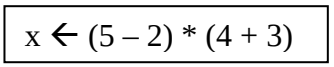
ALTRI ESEMPI



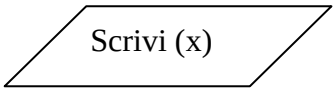
Assegna ad x lo stesso valore di y



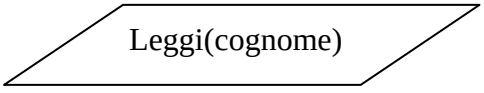
Assegna ad x il valore di y diminuito di 1



Assegna ad x il risultato dell'espressione (5 – 2) * (4 + 3)



Mostra a video il valore della variabile x



Accetta dalla tastiera un valore che memorizzerai nella variabile 'cognome'

I simboli visti fino ad ora servono per istruzioni singole. Vediamo ora come si rappresentano le strutture fondamentali della programmazione (sequenza, selezione, iterazione) ossia le

STRUTTURE DI CONTROLLO

Sequenza

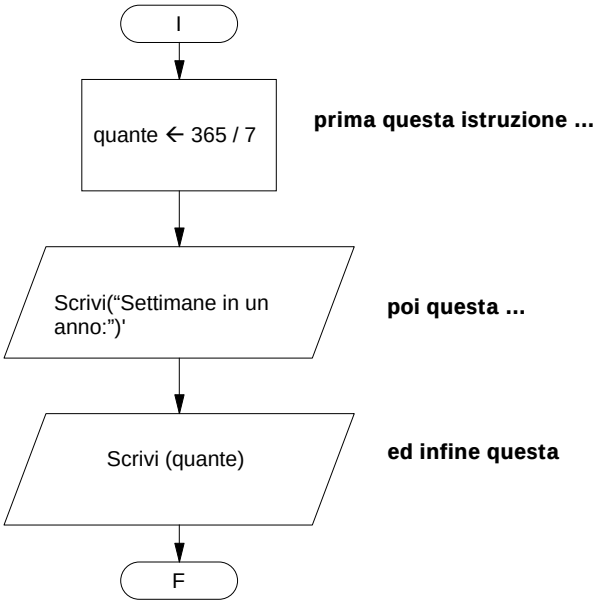
Per indicare che due istruzioni vanno eseguite una dopo l'altra si mettono i loro simboli uno sotto l'altro collegandoli con una freccia. Ecco un esempio.

Calcoliamo quante settimane ci sono in un anno dividendo il numero dei giorni per 7.

Memorizziamo prima il risultato della divisione nella variabile *quante*.

Poi scriviamo un messaggio sul video che 'annuncia' il risultato

Infine scriviamo il risultato, cioè il valore attuale della variabile *quante*.



Il senso della freccia ↓ indica il flusso di esecuzione.

Istruzioni di Selezione (implica una decisione, una scelta, un'alternativa)

Il solo fatto di poter indicare un'istruzione dopo l'altra (sequenza) non ci consentirebbe di sviluppare programma interessanti. Ci sono innumerevoli situazioni in cui è necessario fare un 'controllo' ed agire di conseguenza.

Esempi

Se l'anno è bisestile allora considera febbraio con 29 giorni, altrimenti consideralo con 28.

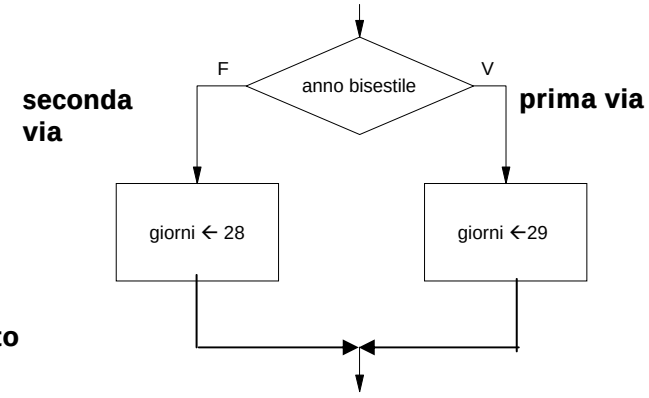
Se l'età è minore di 18 allora applica sconto, altrimenti applica prezzo pieno.

Se la temperatura supera 37 C allora fai suonare l'allarme.

Come avrete osservato, si controlla una *condizione* (anno bisestile ?, età minore di 18 ?, temperatura supera 37 ?) che può risultare vera o falsa. In qualche caso (primi due esempi) ci sono operazioni (diverse) sia nel caso la condizione risulti vera sia nel caso risulti falsa. E' però possibile (terzo esempio) che nel caso la condizione sia falsa non ci sia nulla da fare.

Ecco come si rappresenta in un flow chart la struttura di selezione:

Se l'anno è bisestile
allora
considera febbraio con 29 giorni
altrimenti
consideralo con 28

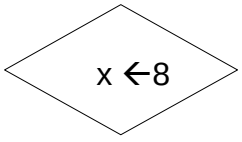


Questo tipo di selezione è detto selezione binaria o a 2 vie

La condizione deve essere scritta all'interno del rombo. Il flusso di esecuzione si divide a seconda del risultato del controllo (V sta per Vero e F sta per Falso) e solo la strada 'a destra' o 'a sinistra' viene percorsa.

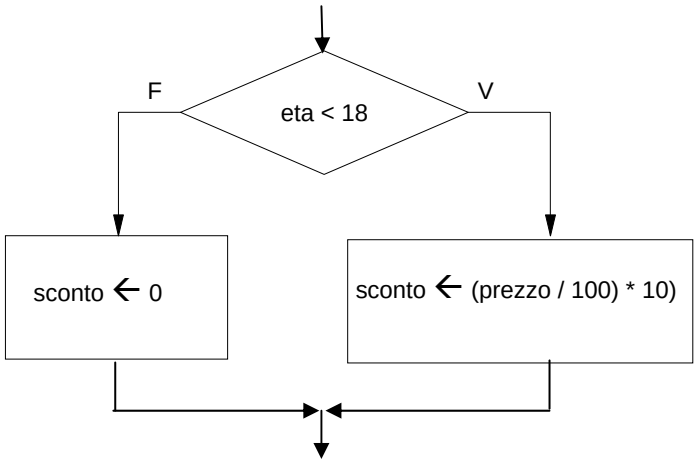
Le istruzioni nella sezione 'vera' (o 'falsa') possono essere anche molte e non una sola come nel nostro esempio. Quando le istruzioni da eseguire a condizione vera (o falsa) sono terminate il flusso si ricongiunge.

Attenzione: nel rombo potete mettere solo condizioni, non assegnazioni! Sarebbe quindi un errore grave il seguente:



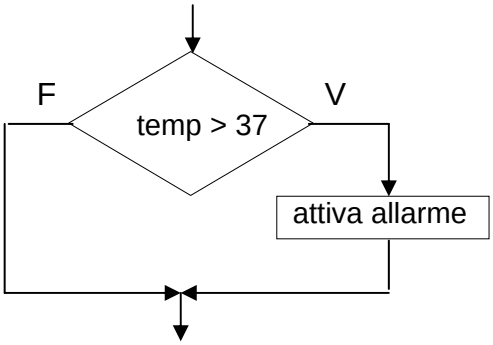
NO !!
ERRORE !!!

Se l'età è minore di 18
allora
applica sconto 10%
altrimenti
applica prezzo pieno.



Nota: nel flow chart immaginiamo che le variabili *eta* e *prezzo* abbiano già un valore valido e che i calcoli indicati siano quindi fattibili. Lo sconto (10%) viene calcolato dividendo il prezzo pieno per 100 (calcolando così l'1%) e moltiplicando il risultato per 10 (ottenendo il 10%).

Se la temperatura supera 37 C
allora
fai suonare l'allarme.

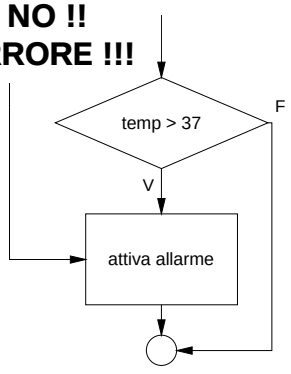


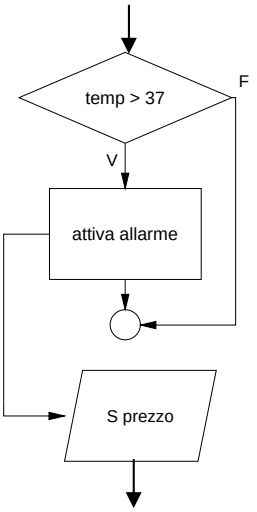
Come potete vedere se non c'è nulla che deve essere fatto quando la condizione è falsa e quindi il flusso si congiunge con la parte 'falso' direttamente all'uscita dell'istruzione.

Questo tipo di selezione è detto selezione unaria o a 1 via

NOTA IMPORTANTE: in ogni diagramma che rappresenta la selezione ci deve essere sempre un solo punto di ingresso (la freccia entrante nel rombo) ed un solo punto di uscita (la freccia uscente dopo il ricongiungimento). Non sarebbe quindi possibile avere flow-chart con frecce che dall'interno escono verso altri punti del diagramma o che dall'esterno giungono in un punto interno:

NO !!
ERRORE !!!



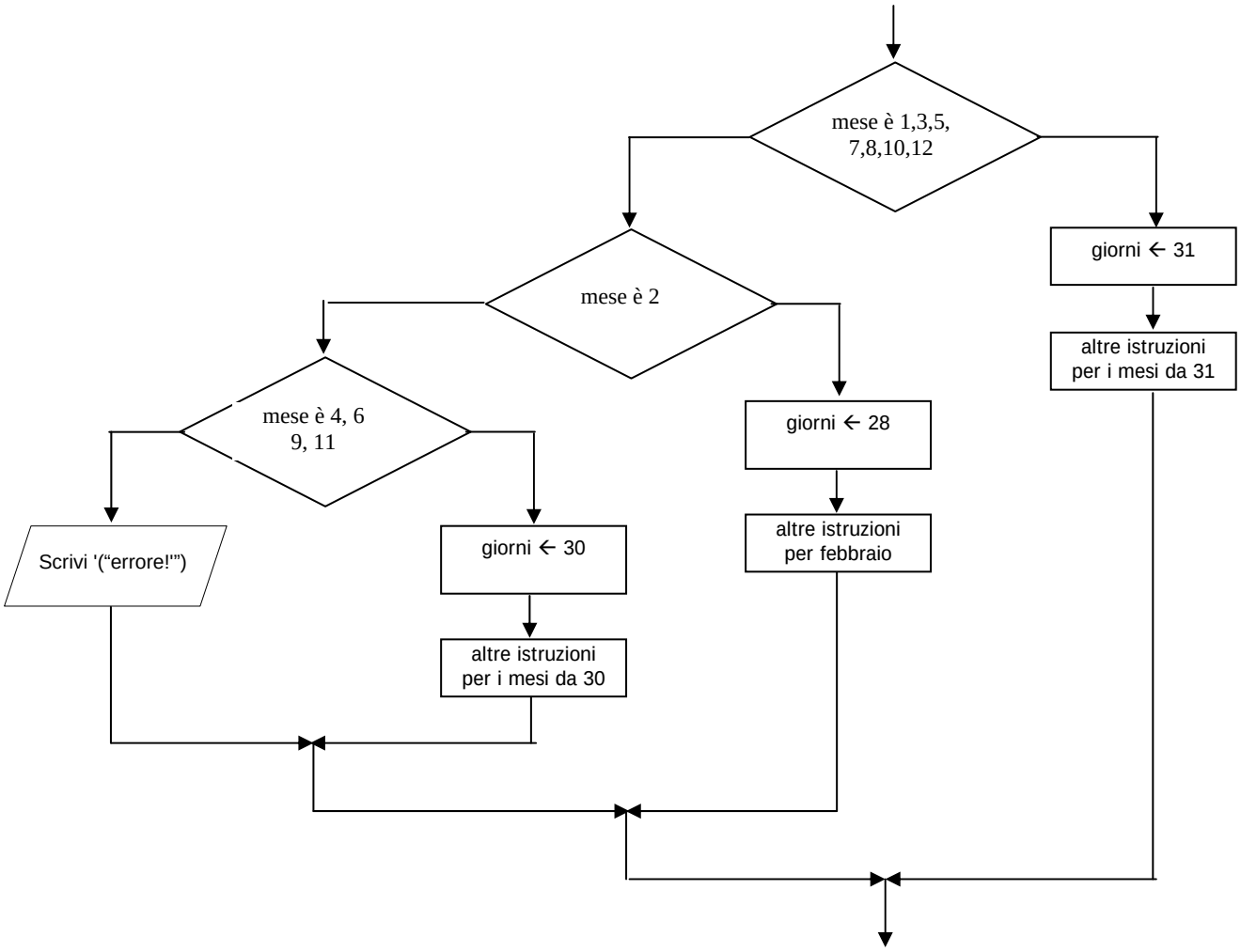


NO !!
ERRORE !!!

Esiste anche un'utile variante che prevede molte vie a seconda del valore di una variabile. Immaginiamo che la variabile *mese* contenga un valore da 1 a 12 che rappresenta uno dei mesi dell'anno. Di nuovo, vorremmo sapere quanti giorni considerare. Vi ricordo che i mesi con 31 giorni sono 1 (gennaio), 3 (marzo), 5 (ecc.), 7, 8, 10, 12; quelli con 30 giorni sono 4 (aprile), 6, 9, 11; immaginiamo in questo esempio, per semplicità, che febbraio (2) ne abbia sempre 28.

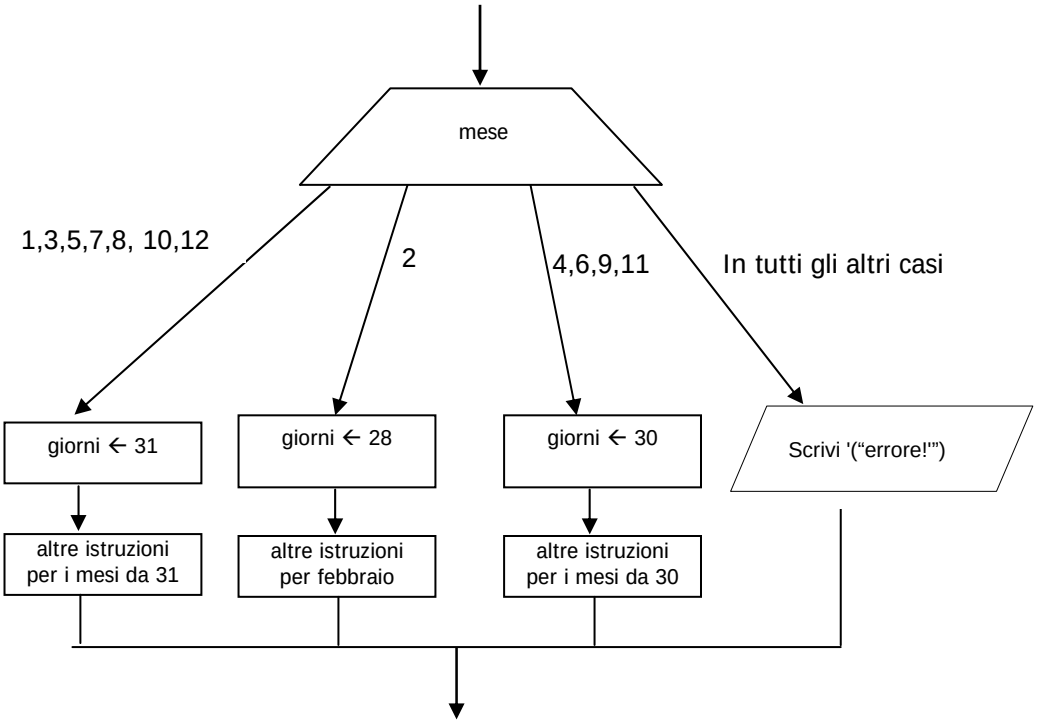
SELEZIONE ENNARIA o AD n VIE

(Flow chart da non utilizzare ma che mette in evidenza la sua traducibilità con la selezione binaria)



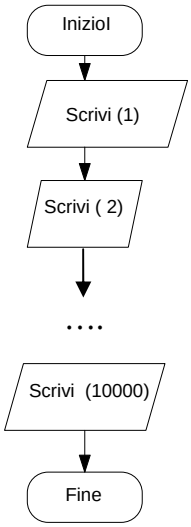
Il flusso può seguire tre vie a seconda del valore della variabile *mese*. I tre flussi possibili si ricongiungono poi nella freccia finale. Anche se non usato in questo esempio, è previsto un flusso da seguire quando non si verifica nessuno dei casi precedenti.

SELEZIONE ENNARIA o AD n VIE
(Flow chart da utilizzare)



Istruzioni di Iterazione (ripetizioni, cicli)

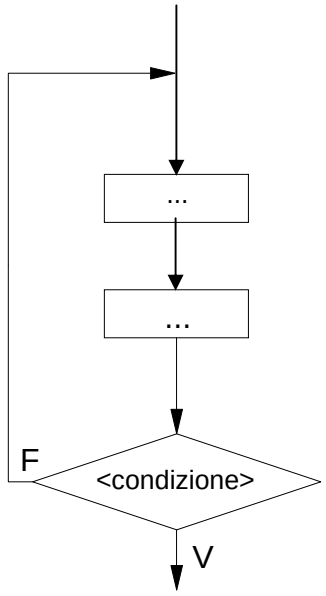
Pur avendo a disposizione sequenza e iterazione rimangono molte le situazioni ‘intrattabili’ con questi strumenti. Consideriamo infatti questo problema all’apparenza molto semplice: stampare i numeri da 1 a 10000. Certamente la soluzione di usare per 10000 volte l’istruzione di scrittura per stampare ogni numero non è molto praticabile



A parte i problemi di trovare uno spazio sufficiente per disegnare schemi così enormi ed il tempo per disegnarli, saremmo costretti a modificare sostanzialmente il diagramma al variare della richiesta (stampare solo i primi 5000 numeri o fino al 15000).

Partendo invece dalla considerazione che si stanno ripetendo istruzioni molto simili (cambia solo in valore da scrivere sul video) si perviene ad una struttura ciclica che fa ripetere una od un gruppo di istruzioni fino al raggiungimento di una condizione (in questo caso il raggiungimento del 10000).

Ecco lo schema generico (cioè non applicato alla soluzione di alcun problema particolare) di un flow chart per la struttura iterativa.



Le istruzioni sono ripetute una prima volta. Giunti alla condizione, se questa è vera allora si esce ed il ciclo termina. Se la condizione è falsa si ritorna all'inizio.

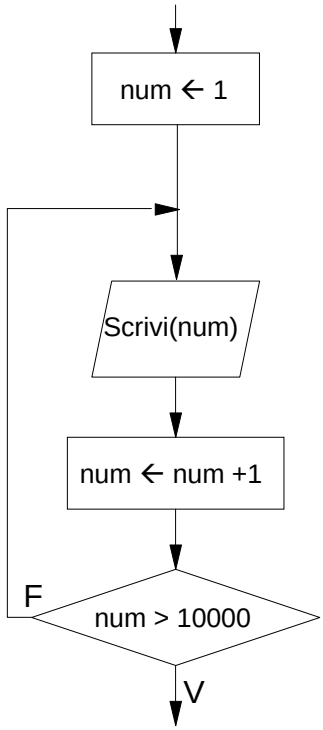
Se all'interno del ciclo, prima o poi, una delle istruzioni farà sì che la condizione diventerà vera, il ciclo terminerà veramente.

Diversamente procederà all'infinito o loop infinito.

Questo tipo di ciclo è detto 'con ripetizione per falso', o a 'uscita per vero' ed ancora 'con controllo in coda' (cioè alla fine del ciclo) o post condizionale

Notiamo ancora: *con questo tipo di ciclo le istruzioni vengono eseguite <u>almeno una volta</u>.*

Vediamo il flow chart completo per la stampa dei numeri da 1 a 10000 che utilizza questa struttura iterativa

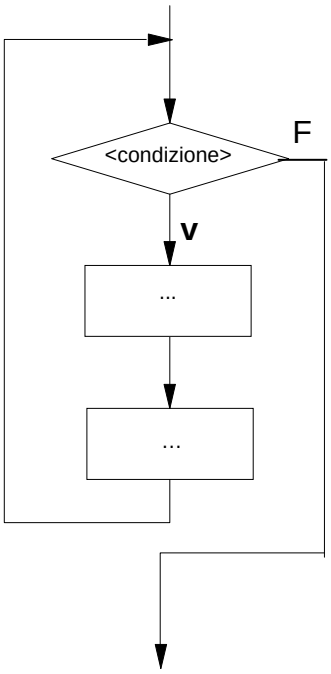


La variabile *num* è usata per controllare l'uscita dal ciclo. Essa viene impostata al valore 1 *prima* di iniziare il ciclo.

Il ciclo inizia stampando sul video il valore della variabile *num* (la prima volta stamperà quindi 1).

La variabile *num* viene incrementata di 1.

Se la variabile *num* ha raggiunto il valore 10001 (quindi ha già stampato il 10000) si esce, altrimenti si ritorna all'inizio (verrà stampato sul video un nuovo numero, si incrementa ancora il valore di *num* e così via …).

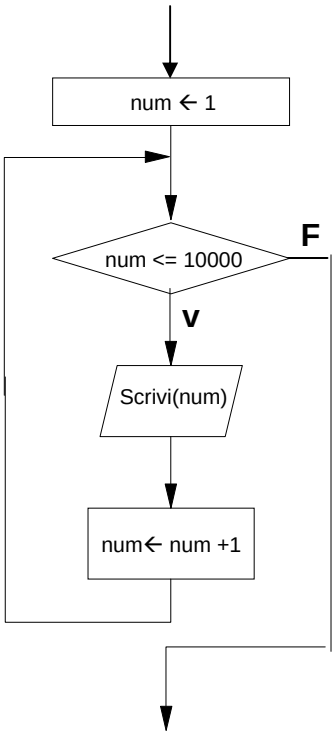


Ed ecco un esempio di flow chart **per l'altro tipo** fondamentale **di ciclo**, **quello detto di 'ripetizione per vero' o ad 'uscita per falso' ed ancora con 'controllo in testa' (cioè all'inizio del ciclo) o precondizionale**

Infatti fintanto che la condizione rimane vera il ciclo viene ripetuto. Non appena la condizione diviene falsa il ciclo termina.

Il ciclo potrebbe anche non cominciare neppure, se la condizione fosse da subito falsa. Il ciclo visto in precedenza, invece, controllando la condizione alla fine, almeno una volta comunque esegue le istruzioni (e questo potrebbe essere indesiderabile quando si vogliono eseguire le istruzioni del ciclo solo se è vera fin da subito una certa condizione).

Vediamo il flow chart completo per la stampa sempre dei numeri da 1 a 10000 che utilizza questa struttura iterativa



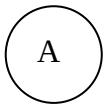
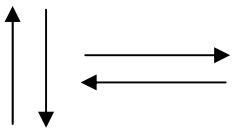
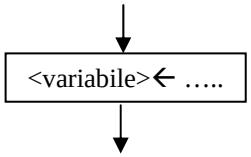
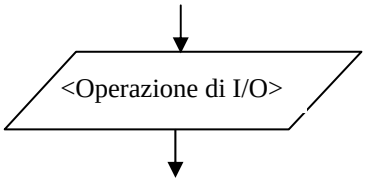
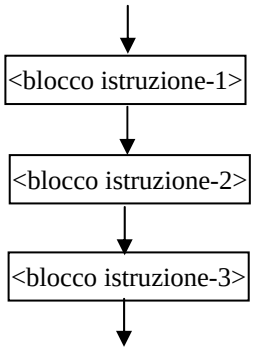
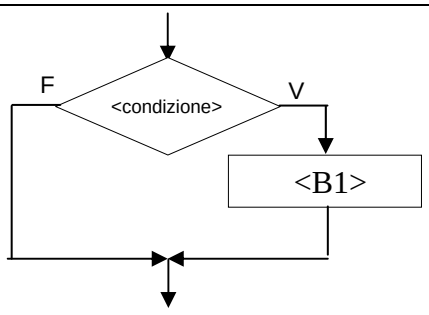
La variabile *num* è usata per controllare l'entrata del ciclo Essa viene impostata al valore 1 *prima* di iniziare il ciclo.

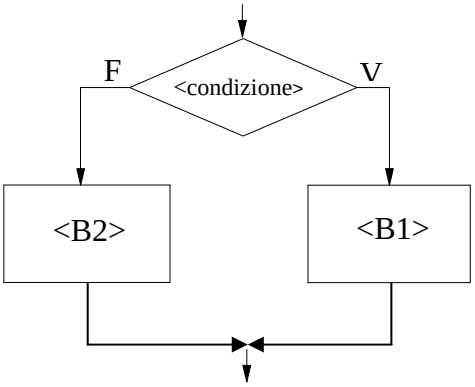
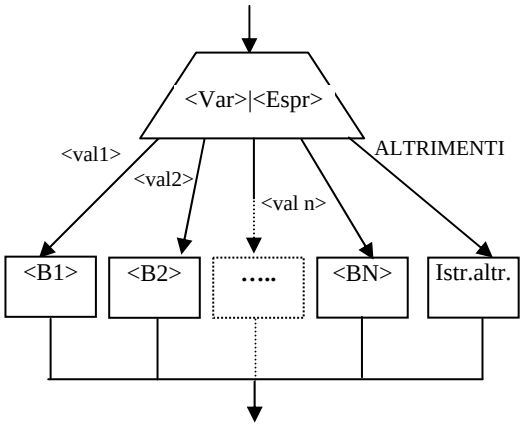
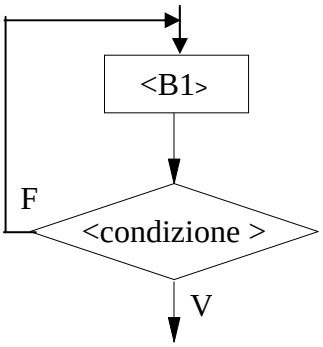
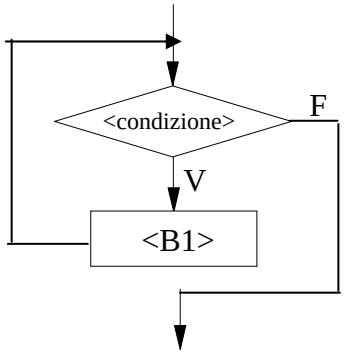
Il ciclo inizia stampando sul video il valore della variabile *num* (la prima volta stamperà quindi 1).

La variabile *num* viene incrementata di 1.

Se la variabile *num* ha raggiunto il valore 10001 (quindi ha già stampato il 10000) si esce, altrimenti si ritorna all'inizio (verrà stampato sul video un nuovo numero, si incrementa ancora il valore di *num* e così via …).

...

Azioni dell'ALGORITMO da rappresentare	PSEUDOISTRUZIONI della PSEUDOCODIFICA	Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI
Inizio	**INIZIO**	
Fine	**FINE**	
Connettore	Indica l'inizio o la fine di una parte di diagramma che è stato diviso per comodità di lettura. Contiene al'interno un simbolo o un numero che corrisponde ad un altro connettore	
Linee di flusso	Indica il flusso dei dati all'interno del processo risolutivo dell'algoritmo	
Istruzione di Assegnazione (ISTRUZIONE OPERATIVA)	<variabile> ← <variabile> | <costante> | <espressione>	
Istruzione di I/O da tastiera o su video (ISTRUZIONE OPERATIVA)	**Leggi** (<variabile>) oppure **Scrivi** (<variabile> | <costante >)	
Costrutto di controllo di tipo "sequenza" (ISTRUZIONE DI CONTROLLO)	…… <blocco istruzioni-1> <blocco istruzioni-2> <blocco istruzioni-3> ……	
Costrutto di controllo di tipo "selezione unaria" (ISTRUZIONE DI CONTROLLO)	**SE** <condizione> **ALLORA** <blocco istruzioni allora> **FINE SE**	

Azioni dell'ALGORITMO da rappresentare	**PSEUDOISTRUZIONI della PSEUDOCODIFICA**	**Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI**
Costrutto di controllo di tipo "selezione binaria" (ISTRUZIONE DI CONTROLLO)	**SE** <condizione> **ALLORA** <B1> **ALTRIMENTI** <B2> **FINE SE**	
Costrutto di controllo di tipo "selezione ennaria" (ISTRUZIONE DI CONTROLLO)	**NEL CASO CHE (**<var>>|<espr>) **SIA** < valore 1> **:** <blocco istruzioni 1> < valore 2> **:** <blocco istruzioni 2> …… : …… < valore N> **:** <blocco istruzioni N> **[ALTRIMENTI** : <blocco istruzioni altri casi>**]** **FINE CASO**	
Costrutto di controllo di tipo iterativo con verifica condizione in coda e ripetizione per FALSO (ISTRUZIONE DI CONTROLLO)	**RIPETI** < B1> **FINCHE' (** <condizione>)	
Costrutto di controllo di tipo iterativo con verifica condizione in testa e ripetizione per VERO (ISTRUZIONE DI CONTROLLO)	**MENTRE** (<condizione>) **ESEGUI** < B1> **FINE MENTRE**	

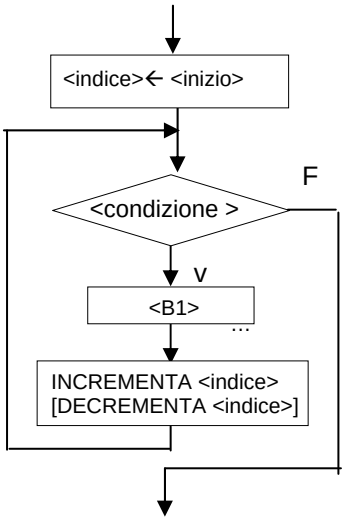
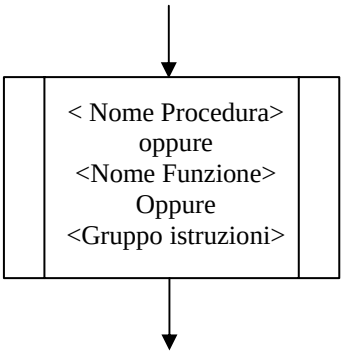
Azioni dell'ALGORITMO da rappresentare	**PSEUDOISTRUZIONI della PSEUDOCODIFICA**	**Simboli del FLOW CHART O del DIAGRAMMA A BLOCCHI**
Costrutto di controllo di tipo iterativo determinato o indicizzato (ISTRUZIONE OPERATIVA)	**PER** <indice> ← <inizio> **[INDIETRO] A** <fine> **ESEGUI** < B1> **INCREMENTA** <indice> **[DECREMENTA** <indice>**]** **FINE PER** Dove <condizione> è: • **<indice> <= <fine>** se il PER è incrementale • **<indice> >= <inizio>** se il PER è decrementale	
Programma principale	**ALGORITMO** <Nome dell'algoritmo> **PROCEDURA** main( ) <seziond dichiarativa locale alla procedura main ( )> **INIZIO** <corpo della procedura main ( )> **FINE**	E' l'insieme di tutti i simboli grafici visti finora
Sottoprogramma o comunque un gruppo di istruzioni da elaborare insieme	**PROCEDURA** <Nome procedura.> **([REF | VAL** <Nome param 1>**:** <Tipo param 1> **,** **REF | VAL** <Nome param 2>**:** <Tipo param 2> , …….., **REF | VAL** <Nome param n>**:** <Tipo param n> **])** < sezione dichiarativa locale alla procedura> **INIZIO** < corpo della procedura> **RITORNA** **FINE** **FUNZIONE** <Nome funzione> **([REF | VAL** <Nome param 1>**:** <Tipo param 1> **,** **REF | VAL** <Nome param 2>**:** <Tipo param 2> , …….., **REF | VAL** <Nome param n>**:** <Tipo param n> **])** **:** < Tipo Risultato> < sezione dichiarativa locale alla funzione> **INIZIO** < corpo della funzione> **RITORNA** <risultato> **FINE**	

TABELLE RIASSUNTIVE PER L'ANALISI DEI DATI

DATI DI INPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile (1)	**Tipo dati (2)**	**Tipo Allocazione (3)**	**Valori ammessi (4)**	**Descrizione (5)**
DATI DI OUTPUT DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**
DATI DI ELABORAZIONE (LAVORO) DEL PROBLEMA PRINCIPALE (PROCEDURA MAIN)				
Nome variabile oppure **nome costante (6)**	**Tipo dati**	**Tipo Allocazione**	**Valori ammessi**	**Descrizione**

(1) Il nome di una variabile deve essere scritto **tutto con caratteri minuscoli** senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di variabili costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di variabili valide i seguenti: *a, pippo, somma-2, somma_2*
Sono nomi di variabili non valide i seguenti: A*, Pippo, somma 2, somma$2*

(2) I tipi utilizzabili sono i tipi di dato semplice (un solo valore possibile per una variabile in un certo istante durante 'esecuzione dell'algoritmo) o tipo di dato strutturato o struttura dati (più valori possibili per una variabile in un certo istante durante l'esecuzione dell'algoritmo)
I tipi di dato semplice che utilizzeremo sono:
INT: per i numeri interi relativi (eventualmente con il segno) ma senza parte decimale; i valori possibili per questo tipo di variabile sono i normali interi numeri scritti nella modalità consueta;
Esempio*: 12 -45 +32*

REAL: per i numeri reali (eventualmente con il segno) con parte decimale; ; i valori possibile per questo tipo di variabile sono i normali numeri reali scritti nella modalità consueta utilizzando la virgola decimale;
Esempio*: 12,34 -45,00 +32,1 ma anche 12 -45 +32*

CHAR: per il singolo carattere alfanumerico secondo la codifica ASCII estesa: i valori possibili per questo tipo di variabile sono i caratteri racchiusi da due apici singoli;
Esempio*: 'a' 'M' '$' '@' ';' 'ƀ'*

BOOL: per il singolo valore di verità dell'algebra booleana: i valori possibili per questo tipo di variabile sono i seguenti: VERO oppure FALSO;

(3) Da valorizzare con:
STAT per indicare l'allocazione della variabile di tipo statico (ossia costante per tutta la durata dell'algoritmo)
DIN per indicare l'allocazione della variabile di tipo dinamico (ossia modificabile nel corso dell'esecuzione dell'algoritmo)
N.B. per questa prima fase useremo variabili esclusivamente STATICHE per le quali, quindi, occorre valorizzare con **STAT** la colonna "Tipo Allocazione" ad eccezione delle costanti per le quali scriveremo **N.A.** ossia non applicabile nella medesima colonna.

(4) In questa colonna occorre indicare, utilizzando se possibile il linguaggio della matematica, le condizioni che i valori di quella variabile devono rispettare nel corso dell'algoritmo.
Esempio*: 'Se a è la variabile STAT di tipo INT che rappresenta il primo coefficiente di un'equazione di secondo grado è ovvio che dovrà essere scritta, nella colonna "Valori ammessi", la condizione* ***a≠0***

(5) Da valorizzare indicando brevemente ma in modo significativo la spiegazione del significato della variabile.

(6) Il nome di una costante deve essere scritto **tutta con caratteri MAIUSCOLI** senza utilizzare spazi e caratteri speciali (quali &, *,#, @). In caso di nomi di costanti costituiti da più di una parola è possibile utilizzare il carattere '-' (trattino) oppure '_' (underscore).
Esempio: Sono nomi di costanti valide i seguenti: *PIGRECO PI-GRECO PI_GRECO*
Sono nomi di costanti non valide i seguenti: *Pigreco PI+GRECO PI&GRECO*