

1.1 INTRODUZIONE ALL'INFORMATICA

L'informatica è:

- *La scienza dei calcolatori elettronici (computer science)*
- *L'insieme delle applicazioni*
- *La scienza dell'informazione (Information theory)*
- *La scienza della rappresentazione e della elaborazione dell'informazione*
- *Scienza dell'artificiale basata su molti livelli di astrazione (progettazione di "mondi artificiali" o "astratti")*

1.2 AREE DISCIPLINARI DELL'INFORMATICA

(secondo ACM – Association for computing machinery)

- Algoritmi e strutture dati
- Linguaggi di programmazione
- Architetture degli elaboratori
- Sistemi operativi
- Ingegneria del software
- Computazione numerica e simbolica
- Basi di dati e sistemi per il reperimento dell'informazione
- Intelligenza artificiale
- Visione e robotica
- Sistemi distribuiti e reti di calcolatori

1.3 MACCHINE DA CALCOLO E COMPUTER

(nella storia)

- *Abaco (2000 a.c.)*
- *Pascal, Leibnitz...meccaniche (1600-1700)*
- *Jacquard, Babbage...a programma (1800)*
- *Hollerith (1900)*
- *Aiken (MARK 1), Mauchly e Eckert (ENIAC)...(1940-1945)*
- *Macchina di Von Neumann...calcolatore elettronico digitale a programma memorizzato (universale)*

1.4 INTRODUZIONE ALL'INFORMATICA 2

1. GENERAZIONI DI COMPUTER

- Valvole (1939)
- Transistor (1947)
- Circuiti integrati (1959) (detti anche chip)
- Microprocessori (1975)
- Parallelismo massiccio

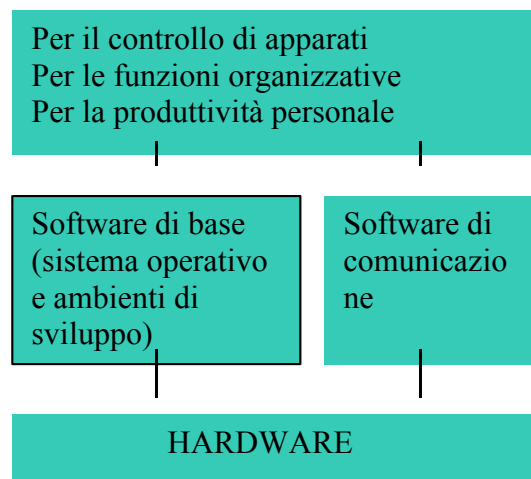
L'evoluzione in campo informatico è iniziata creando calcolatori sequenziali, cioè che svolgono 1 operazione per volta; Poi si è sviluppata con la creazione di calcolatori paralleli, capaci di svolgere molte operazioni contemporaneamente. In questi ultimi si intende "velocità" il n° di operazioni che svolge il calcolatore in una unità di tempo (sec).

2. PUNTI DI VALUTAZIONE PER UN CALCOLATORE

- Tecnologia
- Dimensioni (scala di integrazione)
- Potenza elettrica dissipata
- Velocità di calcolo
- Dimensioni di memoria
- Affidabilità
- Costo

3. DEFINIZIONE DI SISTEMI INFORMATICI

- Sistemi complessi (a vari livelli) hardware e software (e firmware) con molte parti interagenti interfacciati con un ambiente per svolgere un compito in modo autonomo (automatico) o interattivo (con operatori umani)
- **HARDWARE**: è la componente fisica, e quindi soggetta ad usura.
- **SOFTWARE**: è l'insieme di programmi che comandano l'hardware e non sono soggetti a danni fisici; Tuttavia possono contenere errori di progettazione.
- **SOFTWARE APPLICATIVO**:

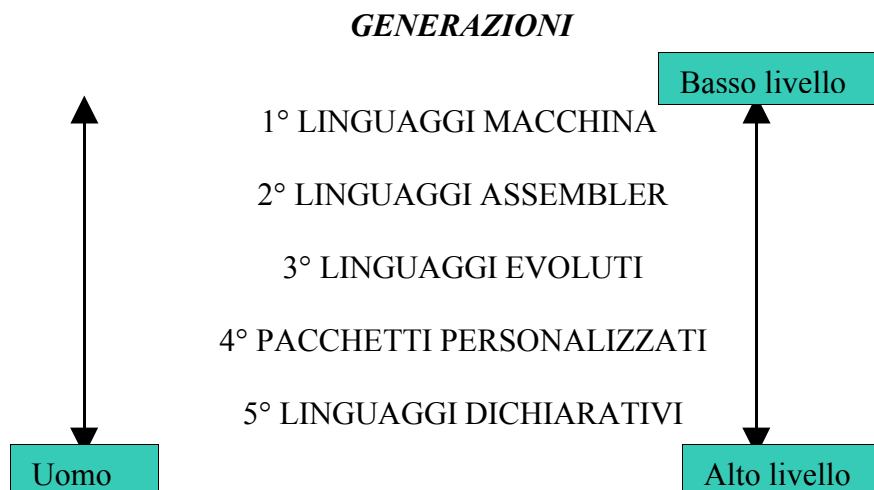


Appunto: La comunicazione tra due terminali può essere sincrona (richiede la risposta del destinatario) asincrona (non richiede la risposta del destinatario) o broadcast (comunicazione con più destinatari).

1.5 LINGUAGGI DI PROGRAMMAZIONE

Il programma determina cosa fare, in che ordine e con che scopo. Ci sono più linguaggi di programmazione (a differenza di quello naturale, il linguaggio artificiale è unificato e scritto con vari programmi).

Di solito i linguaggi di programmazione (circa 2000) sono univoci, mentre la lingua naturale è spesso ambigua.



I linguaggi di programmazione possono essere più vicini e facilitare il compito o all'uomo o alla macchina.

Con l'andare del tempo si è cercato di formulare linguaggi di programmazione più facili per l'uomo che per la macchina; Tuttavia esistono situazioni ove è ancora opportuno usare il linguaggio più vicino alla logica della macchina.

Partendo dagli anni '60 con l'OS (operations system), capace di svolgere più di 1000000 di istruzioni, si è cercato di incrementare sempre di più la produttività dell'elaboratore aumentando il numero di istruzioni svolte dall'elaboratore.

Si sono anche creati programmi traduttori detti "compilatori" che traducono i linguaggi più evoluti a linguaggi macchina.

1.6 PARADIGMI DI LINGUAGGI DI PROGRAMMAZIONE

	Lisp	MC	Scheme	
	Simula	C++	ada 95	Funzionali
	Smalltalk	Java		Oggetti
Linguaggi Macchina	Fortran	Basic	Ada	Imperativi
	Cobol	Algol	Apl	Pascal
		Gpss	Prolog	Dichiarativi

FORTRAN: Per applicazioni scientifiche

COBOL: Linguaggio per applicazioni gestionali

BASIC: Primo linguaggio per PC, più interattivo e di facile utilizzo

PASCAL: E' particolarmente curato nella struttura. Oggi il suo utilizzo è diminuito ma è utile per fissare i concetti della programmazione. Il Pascal può fare parte di Delphi (sistema di programmazione ad oggetti).

1.7 IL CONCETTO DI ALGORITMO

Algoritmo deriva da al-kowarizmi, matematico arabo del IX secolo.

DEFINIZIONE: Metodo GENERALE(*), ASTRATTO(**), EFFETTIVO(***) di risoluzione di problemi formulati esplicitamente.

(*) sommare 3+4 (particolare)

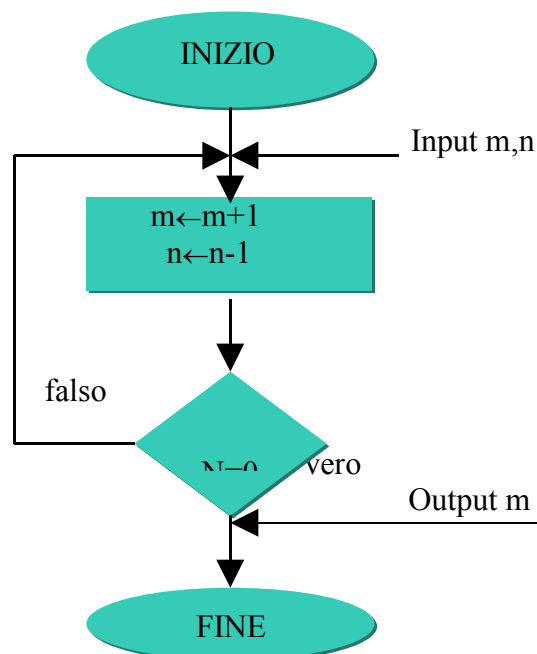
sommare $x+y$ (generale per ogni coppia di x,y)

(**) indipendente del metodo di rappresentazione .

(***) Soddisfa le condizioni di definitezza ed eseguibilità.


1.8 DIVERSE RAPPRESENTAZIONI DELLO STESSO ALGORITMO

Flowchart (diagramma di flusso)



Questo Flow-chart ha lo scopo di sommare $m+n$, con una rappresentazione statica ma dall'andamento dinamico; Il principio con cui vengono sommati i numeri è il seguente:

m	n
7	3
8	2
9	1



(prendo 1 unità da n e la metto in m fino a quando $n=0$)

SPIEGAZIONE DEL DIAGRAMMA:

IN ITALIANO	IN PSEUDOCODICE	Sintassi
1. Acquisisci in ingresso I valori m,n	<u>BEGIN</u> <u>READ</u> m, n	
2. Incremento m di 1 e Decremento n di 1	<u>REPEAT</u> $m \leftarrow m+1$ $n \leftarrow n-1$	
3. Se $n \neq 0$ torna a 2.	<u>UNTIL</u> $n=0$	
4. Produci in uscita il Valore di m	<u>WRITE</u> m <u>END.</u>	

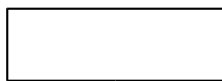
- $m \leftarrow m+1$

Rappresenta il valore della variabile in un istante (sinistra) e in un istante successivo (destra)

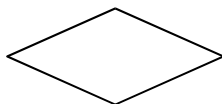
- Assegnamento (assegno a m il valore ottenuto da m+1)
- Variabile = simbolo che rappresenta valori diversi nel tempo
- Il significato delle forme geometriche in un flow-chart è il seguente:



INIZIO o FINE



COMPONENTI DELL'ALGORITMO PER
MODIFICA DELLA VARIABILE



OSSERVAZIONE SULLA VARIABILE
(condizioni): PUO' ESSERE VERA (true)
o FALSA (false)

Quindi il programma è la rappresentazione degli algoritmi in una particolare situazione (anche se l'algoritmo è indipendente dalla situazione).

1.9 L'EFFETTIVITA'

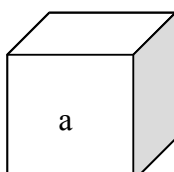
(condizioni di definitezza ed eseguibilità di un algoritmo)

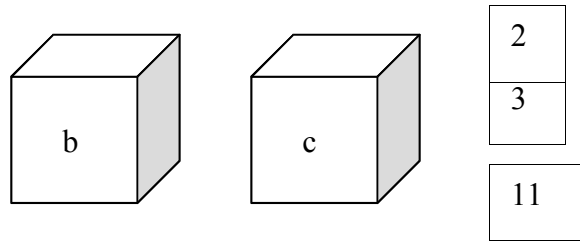
1. Ha zero o più dati in ingresso (quantità assegnate prima dell'inizio)
2. Ha uno o più dati in uscita
3. Deve poter terminare dopo un numero finito e discreto di passi
4. Ogni passo deve essere definito in modo preciso e non ambiguo, per ogni caso possibile
5. Le azioni (operazioni) devono essere eseguite da un esecutore che le sa interpretare, in quantità finita di tempo

1.10 STRUZIONI DI LETTURA

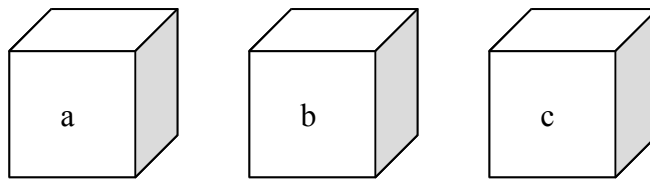
Leggi a,b,c

Assegna alle variabili di nome a,b,c i valori forniti dall'esterno, nell'ordine





Leggi i valori a,b,c



1.11 **VARIABILI e ASSEGNAMENTI**

Una variabile:

- Ha un nome
- Denota un valore

Istruzione di assegnamento:

esempio: $A \leftarrow B + C$

$\text{sin} \leftarrow \text{dest}$ N.B.: 1 sola variabile

Calcola il valore dell'espressione $B+C$ (sommando il valore della variabile B con il valore della variabile C) e lo assegna come (nuovo) valore della variabile A.

N.B.: Il valore di A viene modificato mentre quelli di B e C sono fissi.

Istruzione di scrittura:

Un valore si può usare una sola volta:

Es. $A \leftarrow B$
 $B \leftarrow A$ } SBAGLIATO

$A \leftarrow B$
 $B \leftarrow C$
 $C \leftarrow A$ } GIUSTO (rinomino la B in C per poi riutilizzarla con A)

Esempio: Stampa x1,x2

Stampa i valori delle variabili di nome x1 e x2

Nome x1 e x2

Stampa "nessuna soluzione"

Stampa (senza alcuna elaborazione) la sequenza

Tra le virgole.

1.12 **L'ALGORITMO PIU' ANTICO**

Si tratta dell'algoritmo di EUCLIDE [elementi, libro VII, proposizioni I e II, 330-310 a.c.]

Dati due interi positivi m e n , trovare il loro M.C.D., cioè il minimo intero positivo che divide senza resto sia m che n .

E1.[trovare il resto]:dividere m per n e chiamare r il resto

E2.[è zero?]:se $r=0$ l'algoritmo termina e n è la risposta

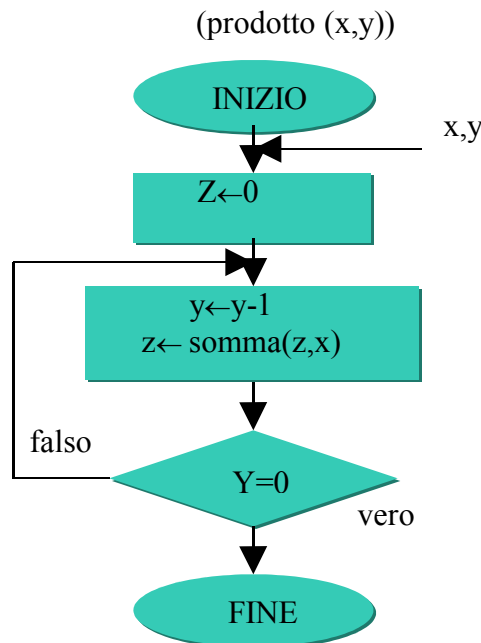
E3.[scambiare]: Porre $m \leftarrow n$ $n \leftarrow r$. Tornare indietro al passo E1.

- (1) m e n sono dati d'ingresso, presi dall'insieme degli interi positivi
- (2) n , dato d'uscita, è il M.C.D. dei dati d'ingresso
- (3) r decresce dopo ogni applicazione del passo E1, e una sequenza decrescente di interi positivi deve necessariamente finire con 0
- (4) La divisione e il resto sono definiti matematicamente e sono eseguibili in un tempo finito.

Proprietà del M.C.D.:

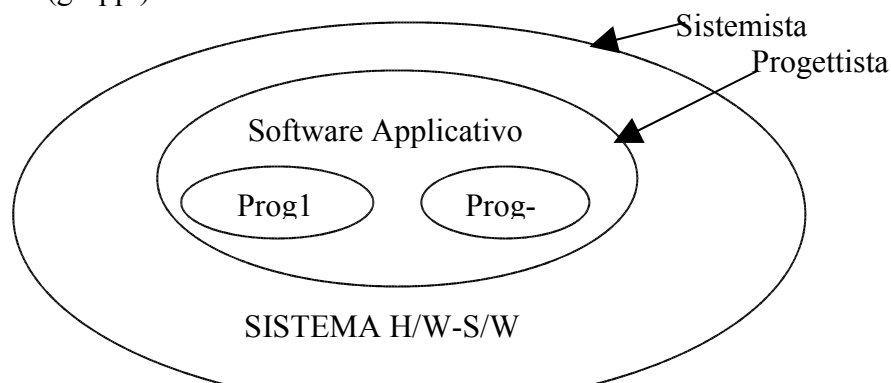
- (1) $M.C.D.(n,m)=M.C.D.(m,n)$
- (2) Se $m=n$ allora $M.C.D.(m,n)=n=m$
- (3) Se $m>n$ e $n=0$ allora $M.C.D.(m,0)=m$
- (4) Se $m>n$ e $n>0$ allora $M.C.D.(m,n)=M.C.D.(n-m,n)=M.C.D.(m-2*n,n)=M.C.D.(m-q*n,n)=M.C.D.(r,n)=M.C.D.(n,r)$

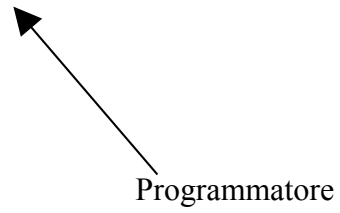
1.13 ALGORITMI CHE USANO ALTRI ALGORITMI



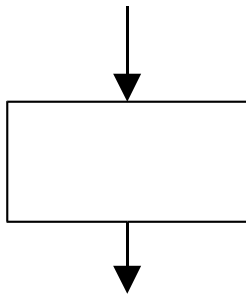
1.14 PROGRAMMAZIONE

- In piccolo (singoli): PROBLEMA → ALGORITMO → PROGRAMMA
- In grande (gruppi):

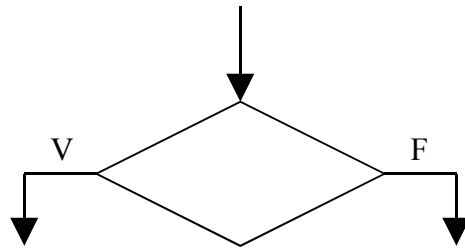




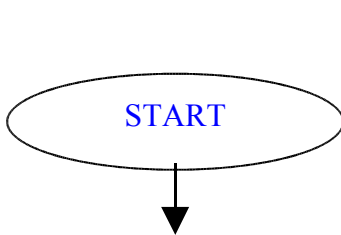
1.15 FLOW-CHART (schemi)



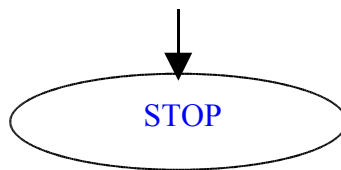
Nodo Operativo
(istruzioni)



Nodo decisionale
(condizioni)

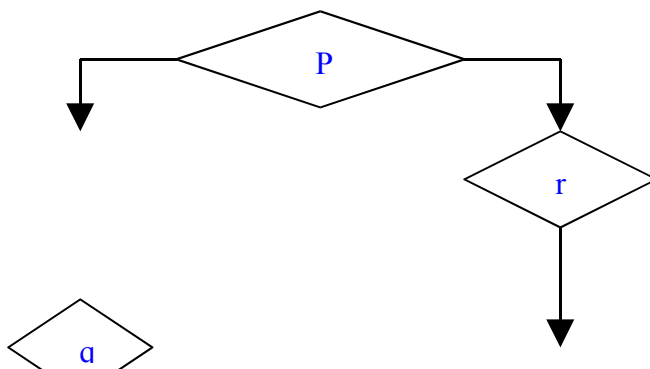


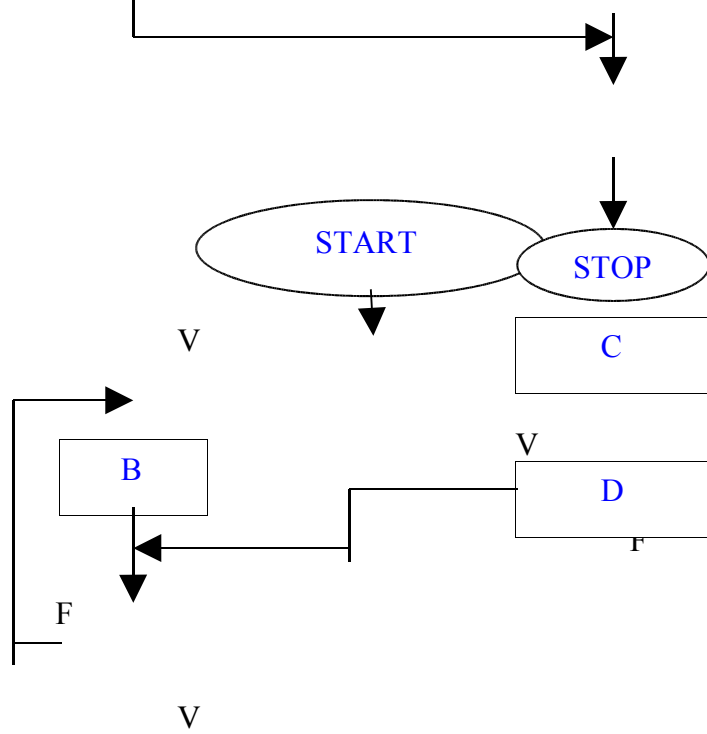
Nodo iniziale



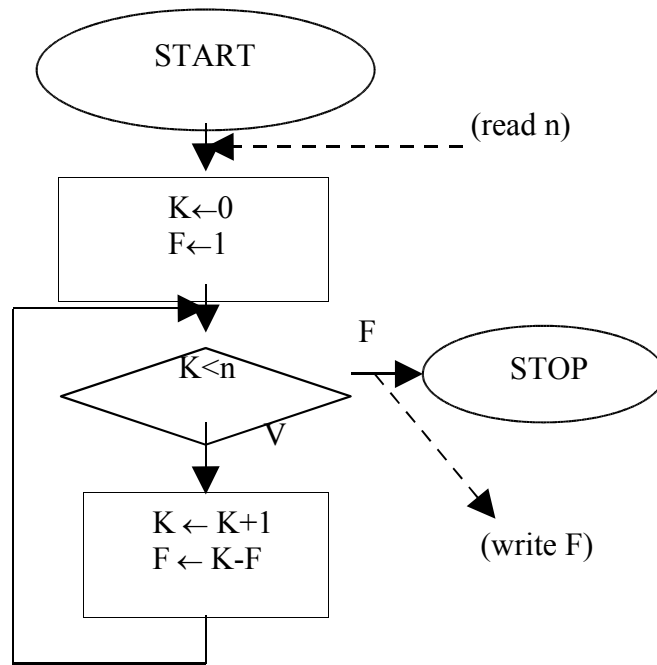
Nodo finale

Esempio:

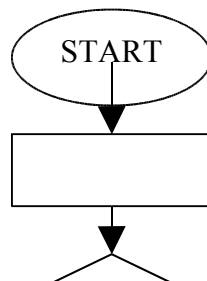


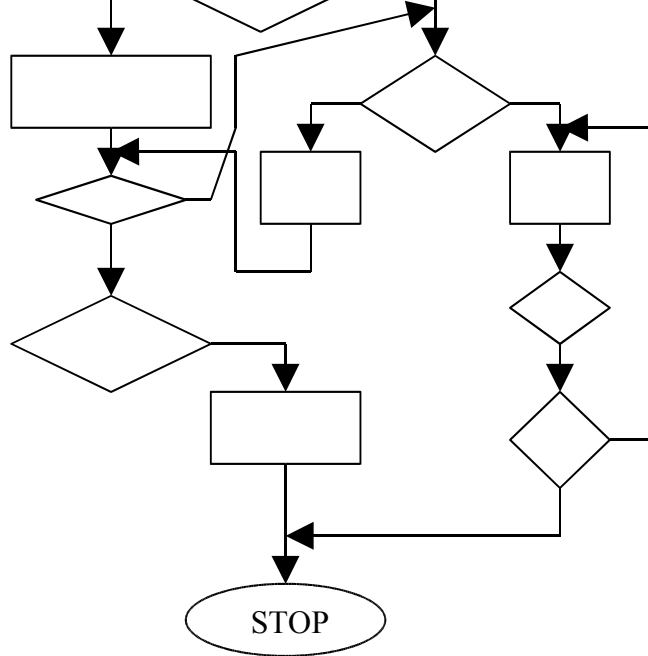


Esempio n°2: CALCOLO DEL FATTORIALE DI $n!$



Esempio di spaghetti-charts:





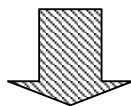
1.16 STILI DI PROGRAMMAZIONE e STRUTTURE DI CONTROLLO

Per costruire un programma funzionante e ben strutturato si devono osservare queste prime regole base:

- Suddividere il programma in moduli coerenti;
- Organizzare il programma in modo da evidenziare la struttura;
- Scrivere commenti significativi per spiegare il programma;
- Usare nomi significativi per procedure e variabili.

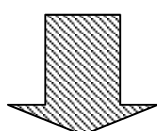
STRUTTURE DI CONTROLLO:

Problema: Programmi scritti con GO TO sono complessi (difficili da modif.);



CERCARE COSTRUTTI LINGUISTICI ALTERNATIVI

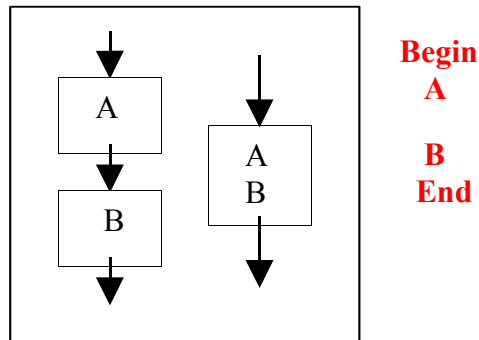
Obbiettivi: Evitare la complessità dei programmi mediante criteri di divisione in parti.



STUDIO DEGLI SCHEMI DI CONTROLLO

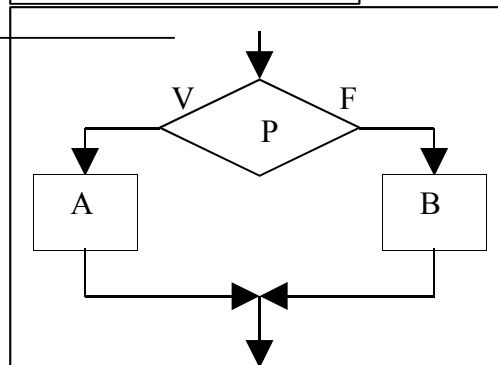
TRE SCHEMI BASE:

- SEQUENZA:

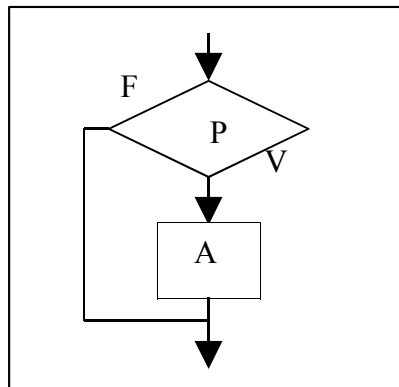


Begin
A
B
End

- SELEZIONE:

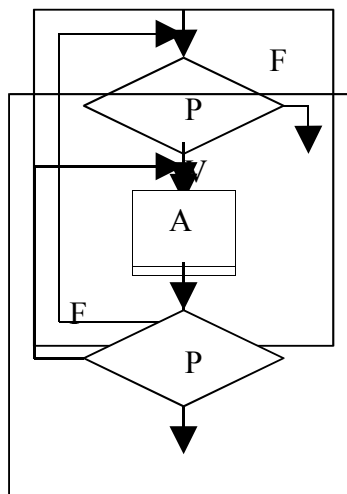


If P
Then A
Else B



If P
Then A

- INTERAZIONE:



While P do A
(Quando P è vera vai in A)

Repeat A until P
(Ripeti A fino a quando P
Diventa vera)

V

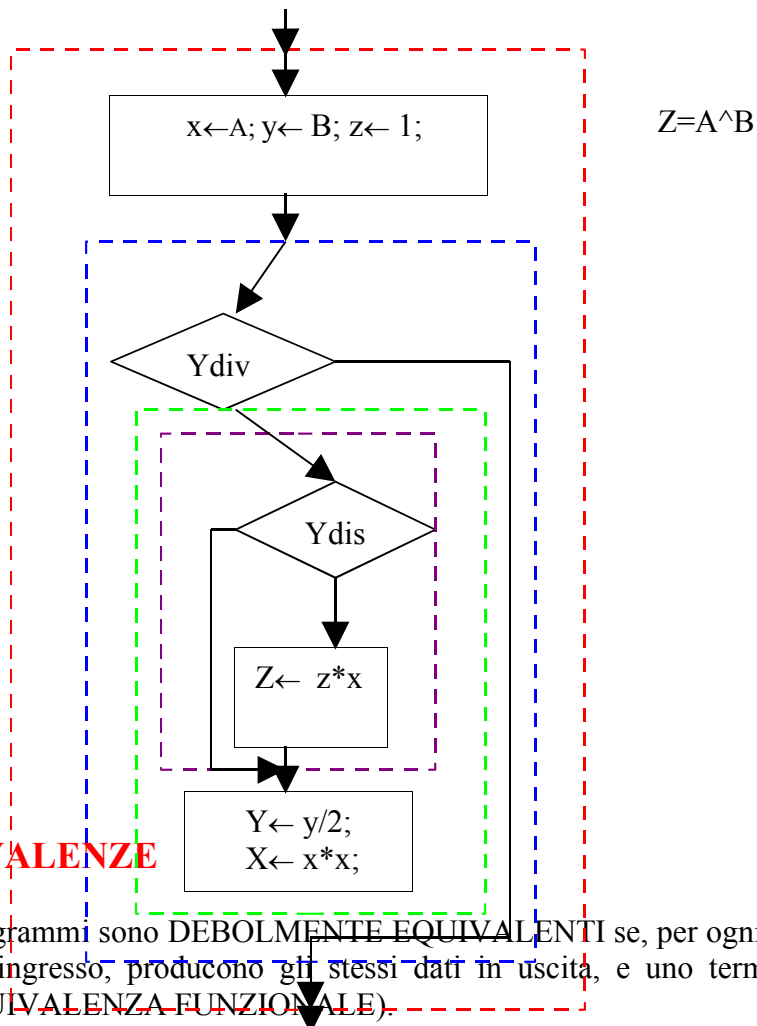
SCHEMA IN PAROLE CHIAVE –NASTING-(nidificato)

```
IF C1 THEN
  IF C2 THEN
    B1
  ELSE
    B2
  ELSE
    IF C3 THEN B3
```

Sequenza di esecuzione:

C1 vera	C1 vera	C1 falsa	C1 falsa
C2 vera	C2 falsa	C3 vera	C3 falsa
B1	B2	B3	-

1.17 COMPOSIZIONE DI STRUTTURE

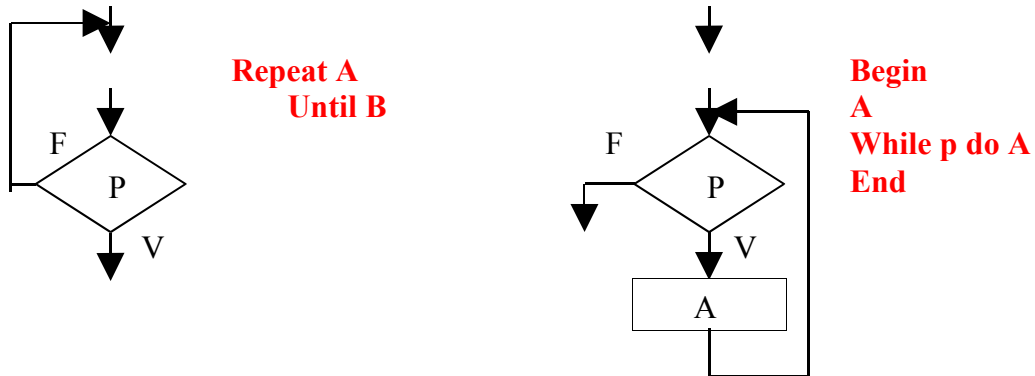


3 2.1 EQUIVALENZE

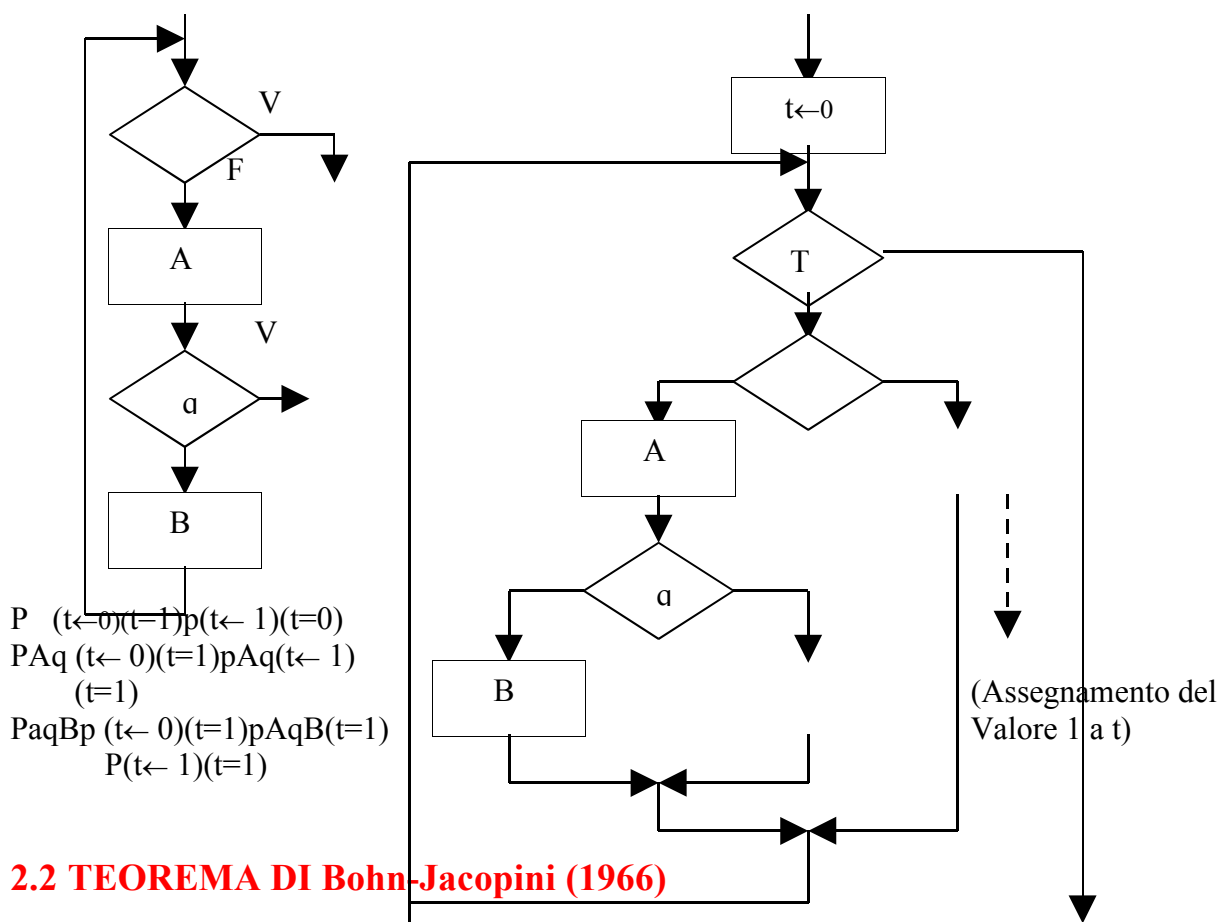
DEBOLE: Due programmi sono DEBOLMENTE EQUIVALENTI se, per ogni stesso insieme di dati in ingresso, producono gli stessi dati in uscita, e uno termina se termina l'altro. (EQUIVALENZA FUNZIONALE).

FORTE: Due programmi sono FORTEMENTE EQUIVALENTI se, per ogni insieme di dati di ingresso, le rispettive sequenze di esecuzione sono uguali. (EQUIVALENZA ALGORITMICA).

ESEMPIO DI PROGRAMMI FORTEMENTE EQUIVALENTI:



ESEMPIO DI PROGRAMMI DEBOLMENTE EQUIVALENTI:



2.2 TEOREMA DI Bohn-Jacopini (1966)

Dato un programma costruito a partire da uno schema flow-chart esiste sempre un programma costruito con gli schemi di base debolmente equivalenti a quello dato.
N.B.: Con i tre schemi si può comporre qualunque funzione ma NON rappresentare qualunque algoritmo.

2.3 DEFINIZIONE DI GRAMMATICA DI UN LINGUAGGIO

$G=[t,n,p,s]$

T = Insieme di simboli terminali

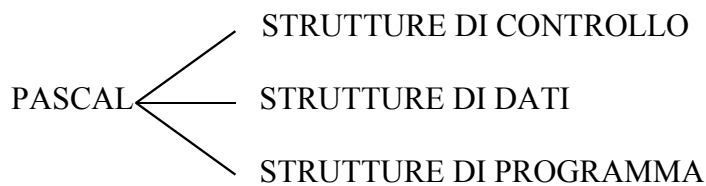
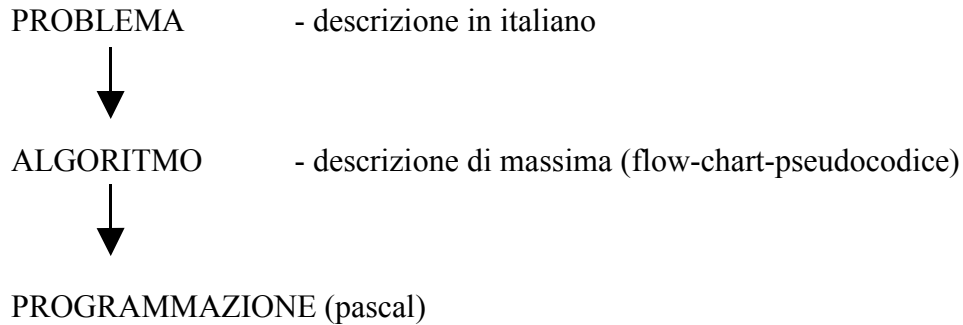
N = Insieme di simboli non terminali (cat. Sintattica)

P = Insieme di regole grammaticali

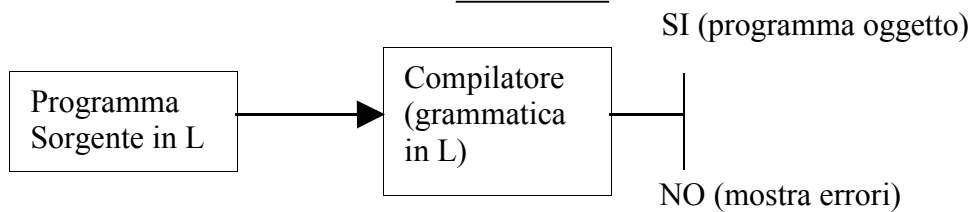
S = Simbolo iniziale (radice, scopo)

Esempio: T unito N = ins.vuoto S appartiene N

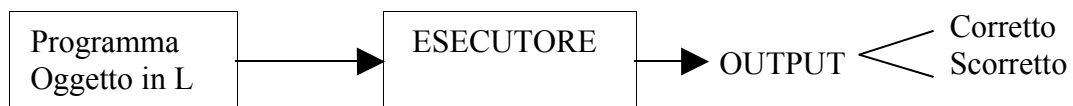
2.4 PROGRAMMAZIONE STRUTTURATA



SINTASSI:

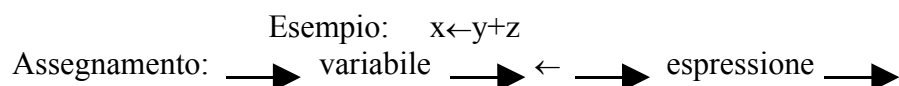


SEMANTICA:



DESCRIZIONE DEI LINGUAGGI DI PROGRAMMAZIONE:

SINTASSI: carte sintattiche



SEMANTICA: a parole

Viene calcolato il valore dell'espressione in funzione del valore corrente delle variabili componenti e il risultato diventa il nuovo valore delle variabili.

3.1 IL PASCAL

La storia:

- Progettato nel 1968-1969
- Realizzato nel 1970
- Pubblicato nel 1971
- Definizione assiomatica nel 1973
- Rapporto rivisto e manuale nel 1974

Gli obiettivi:

- Fondatezza concettuale
- Facilità di apprendimento
- Compilazione a una passata

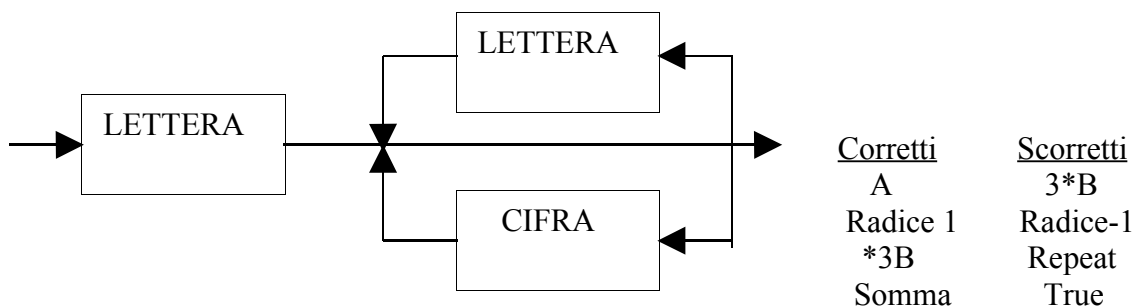
L'alfabeto:

- Lettere (A,B,C;a,b,c;)
- Cifre (0,1,2,3)
- Caratteri speciali (+ - * / \uparrow = < > () [] { } . , : ;)
- Carattere 'spazio' o blank (b)

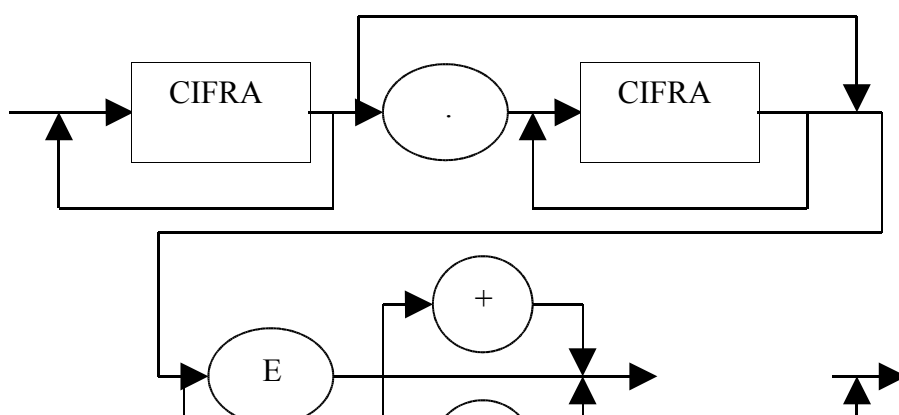
Il vocabolario:

- Simboli speciali (delimitatori) (+ := < > <= >= ..)
- Parole riservate (if, then, else, while, repeat.....)
- Identificatori
- Numeri
- Stringhe

IDENTIFICATORI:



NUMERI:



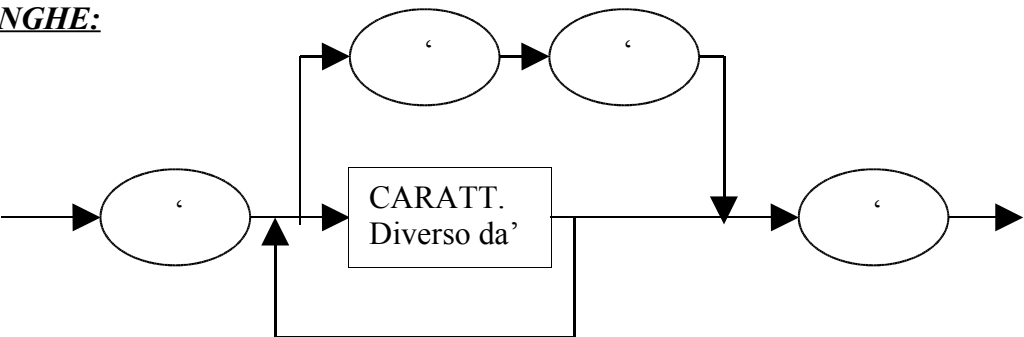


CIFRA

Esempio:

<u>Corretti</u>	<u>Scorretti</u>
3	3,1
03	0.1.4
6272844	XII

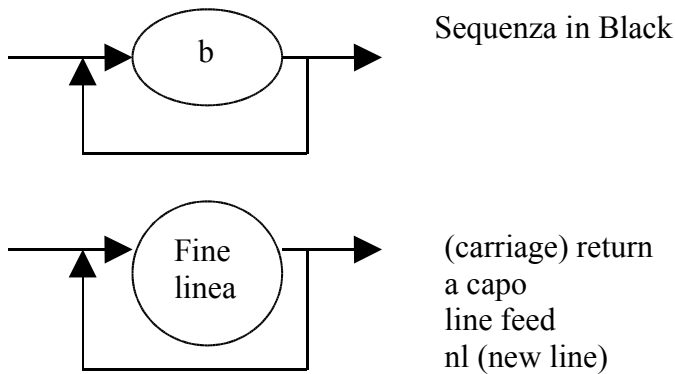
STRINGHE:



Esempio:

<u>Corrette</u>	<u>Scorrette</u>
'A'	'questa e' scorretta'
'X=25'	
'questa e'' corretta'	

SEPARATORI:

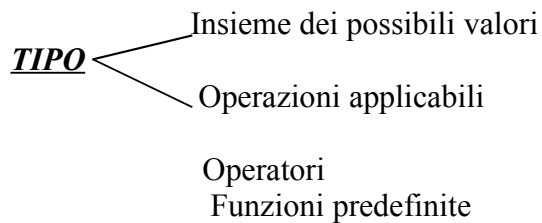


3.2 OGGETTI DI PASCAL

- Variabili
- Costanti
- Espressioni
- Funzioni

Tipi e variabili:

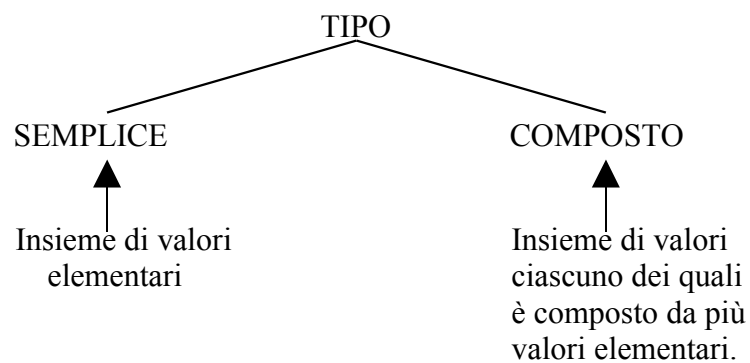
Caratteristiche:



RAPPRESENTAZIONE

FUNZIONALITA'

Tipi semplici e composti:



Esempio:

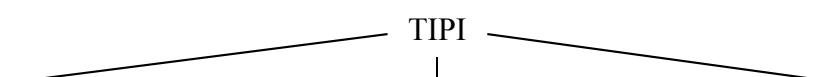
Integer:
N° posti al cinema

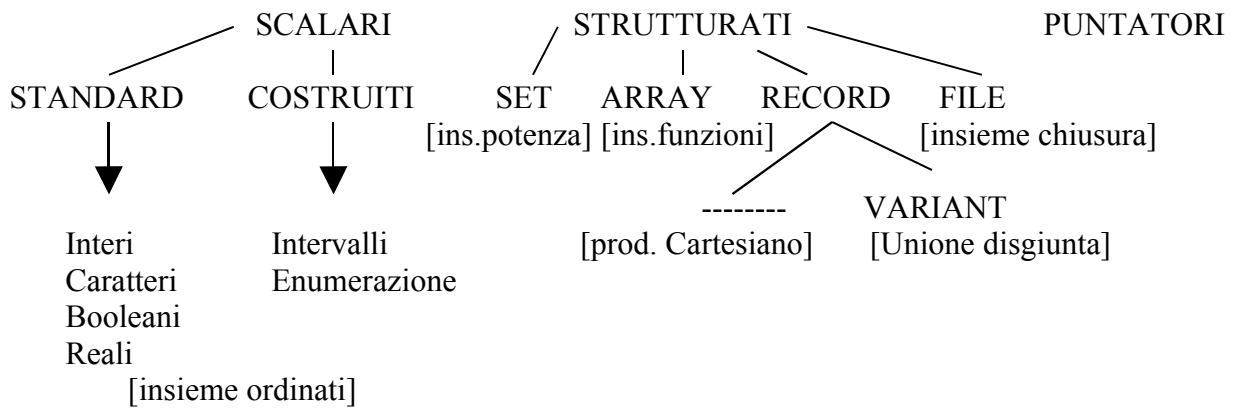
Data:
giorno(integer)
Mese (integer)
Anno (integer)

Scopo dei tipi:

- Esplicitare gli attributi delle variabili
- Controllare l'applicazione degli operatori
- Ottimizzare l'allocazione in memoria
- Definire le regole di composizione dei dati (analogamente alla composizione delle azioni)

Classificazione dei tipi:





3.3 TIPI SCALARI

Insieme finito di valori ordinati

$T = (C_1, C_2, \dots, C_n)$ con $C_i (1 \leq i \leq n)$ identificatori di costanti ed $n = \text{cardinalità di } T$

Assiomi: $C_i \text{ diverso } C_j$ per $i \text{ diverso } j$
 $C_i < C_j$ per $1 \leq i < j \leq n$

Operatori di relazione: $<, \leq, >, \geq, =, \neq$

Funzioni:

$SUCC(C_i) = C_{i+1} \quad (1 \leq i < n)$
 $PRED(C_i) = C_{i-1} \quad (1 < i \leq n)$
 $ORD(C_i) = i \quad (1 \leq i \leq n)$

Tipi enumerativi:

Esempio:

Type mese(gen, feb, mar, ecc...)
 Colore(rosa, giallo, nero, ecc...)
 Sesso(maschio, femmina)

I linguaggi di enumerazione sono ordinati:

esempio: Feb < mar

Devono essere disgiunti:

esempio:

Type mezzi di trasporto(auto, nave, aereo, treno)
 Mezzi terrestri(tram, treno)
 (non è corretto perché non è valutabile auto < treno)

Tipi e variabili:

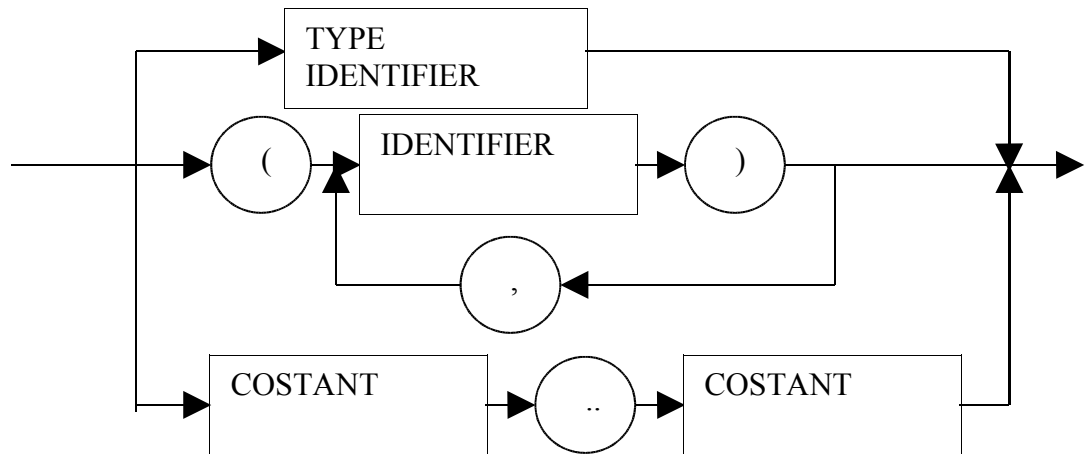
TYPE mese(...)

Definisce l'attributo mese associabile ad una variabile

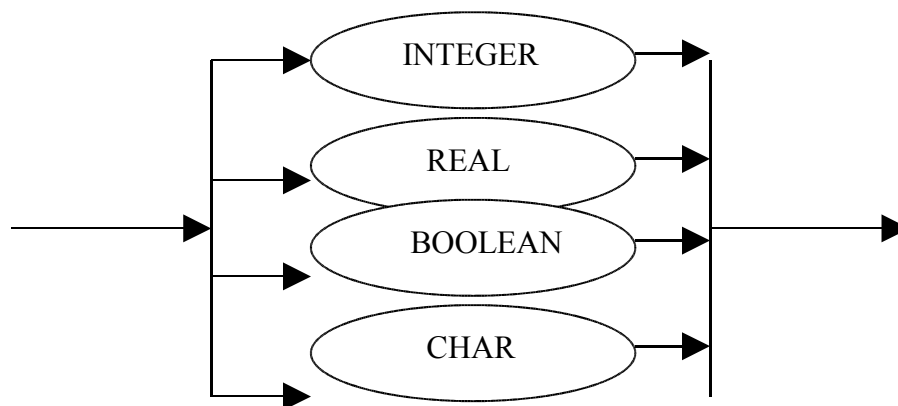
VAR x,y: mese

Dichiara che le variabili x e y sono di tipo mese.
 Esempio: DICHIARAZIONI: var z: integer
 ISTRUZIONI z:=z+1 z ← z+1
 X:=gen ecc..

SIMPLE TYPE:



TYPE IDENTIFIER:



CONGRUENZA DI TIPO (negli assegnamenti):

VAR s:(positivo, negativo)
 VAR C1,C2(rosso, blu, giallo)

S:= positivo	}	CORRETTI
C1:= rosso		
C2:= C1		
C1:= positivo	}	ERRATI
C2:= verde		
C1:= S		

TIPO SUBRAGE:

Si indicano gli estremi di un intervallo
 TYPE Mesestivo= GIU..SET
 Nota:

VAR a,b: Mese
 VAR x,y: Mesestivo
 a:= gen
 b:= ago
 x:= a (può dare degli errori in esecuzione)
 b:= y (và sempre bene)

3.4 TIPI STANDARD

TIPO STANDARD INTEGER

Integer = (-maxint , 1..maxint)
 (maxint: costante dipendente dell'implementazione)

OPERATORI: + , - , * , div , mod
 Esempio: a = ((a div b)*b)+(a mod b)

FUNZIONI: odd(dispari) , abs , sqr(quadrato), pred , succ

TIPO STANDARD REAL

Real = (-10^{e1}..10^{-e2} , 10^{-e3}..10^{e4})
 (e1,e2,e3,e4 sono costanti dipendenti dall'implementazione)
 N.B.: Floating Point = Virgola Mobile

<u>Notazione Decimale</u>			<u>Notazione Scientifica</u>	
3.14	5E6	≡	5*10 ⁶	≡ 5000000
-25.0	-6.0E-4	≡	6*10 ⁻⁴	≡ -0.0006
+0.19	+6.08E+27			

OPERATORI: + , - , * , / , < , >

FUNZIONI: abs , sqr , exp , ln , sin , cos , arctan , trunc , round

TIPO STANDARD BOOLEAN

Boolean = (false, true)

AB	not A	A and B	A or B
FF	T	F	F
FT	T	F	T
TF	F	F	T
TT	F	T	T

OPERATORI: not , and , or

Esempio:
 VAR log : boolean
 Int,a,b,c,d : integer;
 Log:= true
 Log:= int >=0
 If log then....
 If (a<b) and (not(c=d)) then....

TIPO STANDARD CHAR

Char = (..., '\$', '%', '0', ..., '9', 'a', ..., 'z', ...)

FUNZIONI: $\text{chr}(36) = \text{'\$'}$ (in codice ASCII, nel quale ad ogni carattere è associato un valore numerico)
 $\text{Ord}(\text{'\$'}) = 36$



Dà il codice numerico del carattere

$\text{Chr}(\text{ord}(\text{car})) = \text{car}$

$\text{Ord}(\text{chr}(\text{int})) = \text{int}$

+ operatori e funzioni degli scalari (es. $\text{succ}(\text{'L'}) = (\text{'M'})$)

3.5 OPERATORI

Sono definiti $\left\{ \begin{array}{l} \text{DOMINIO (tipo di operandi)} \\ \text{CODOMINIO (tipo di risultato)} \end{array} \right\}$ di ogni operatore e funzione

Operatore o Funzione	1° Operando	2° Operando	Risultato
div	integer	integer	integer
/	real	real	real
not	boolean		boolean
and	boolean	boolean	boolean
trunc	real		integer
chr	integer		char
+	integer o real	Integer o real	integer o re
<	scalare	scalare	boolean
pred	scalare(no real)		scalare ()

Simbolicamente:

$F : D \rightarrow C$ (dominio \rightarrow codominio)

$D1 * D2 \rightarrow C$ (in generale $D1 * \dots * Dn \rightarrow C$)

ESPRESSIONI CON OPERATORI

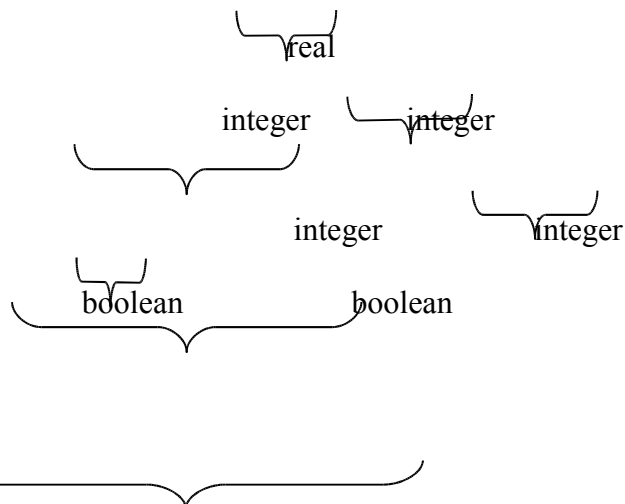
VAR b: boolean

C: char

I, j: integer

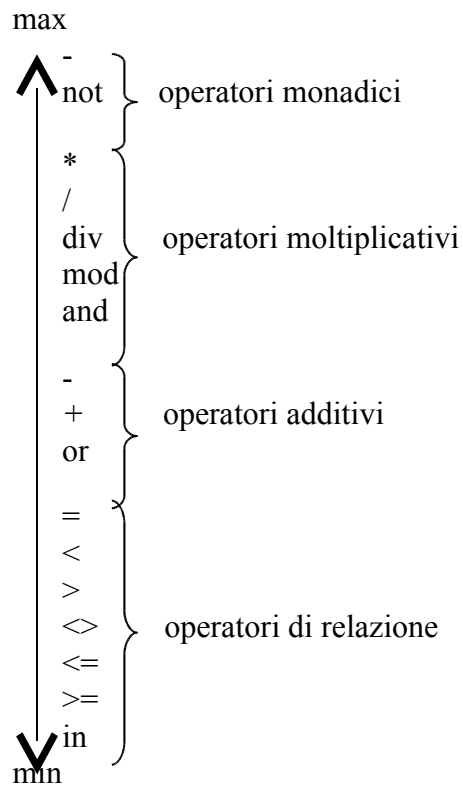
X: real

$B \text{ and } (\text{trunc}(\sin(x)) + \text{ord}(C) < i \text{ div } j)$



boolean

PRECEDENZE DEGLI OPERATORI (4 livelli)

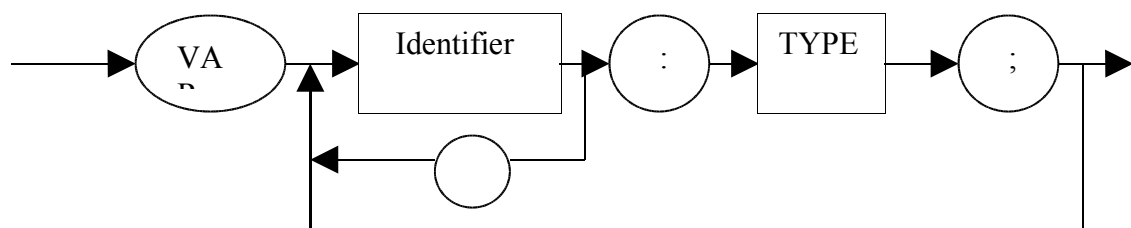


N.B.: a parità di livello ordine da sinistra a destra

3.6 LE VARIABILI

Le variabili sono dei valori dichiarati nella sezione VAR che possono modificarsi con il procedere del programma.

Variable Declaration:



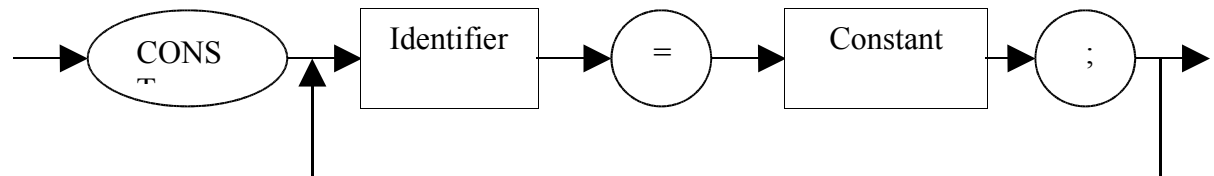
Esempio:

```
VAR x : integer;  
    y,z : real;
```

3.7 LE COSTANTI

Le costanti sono dei valori dichiarati nella sezione CONST che assumono un valore in tale sezione e questo non cambia con lo svolgersi del programma.

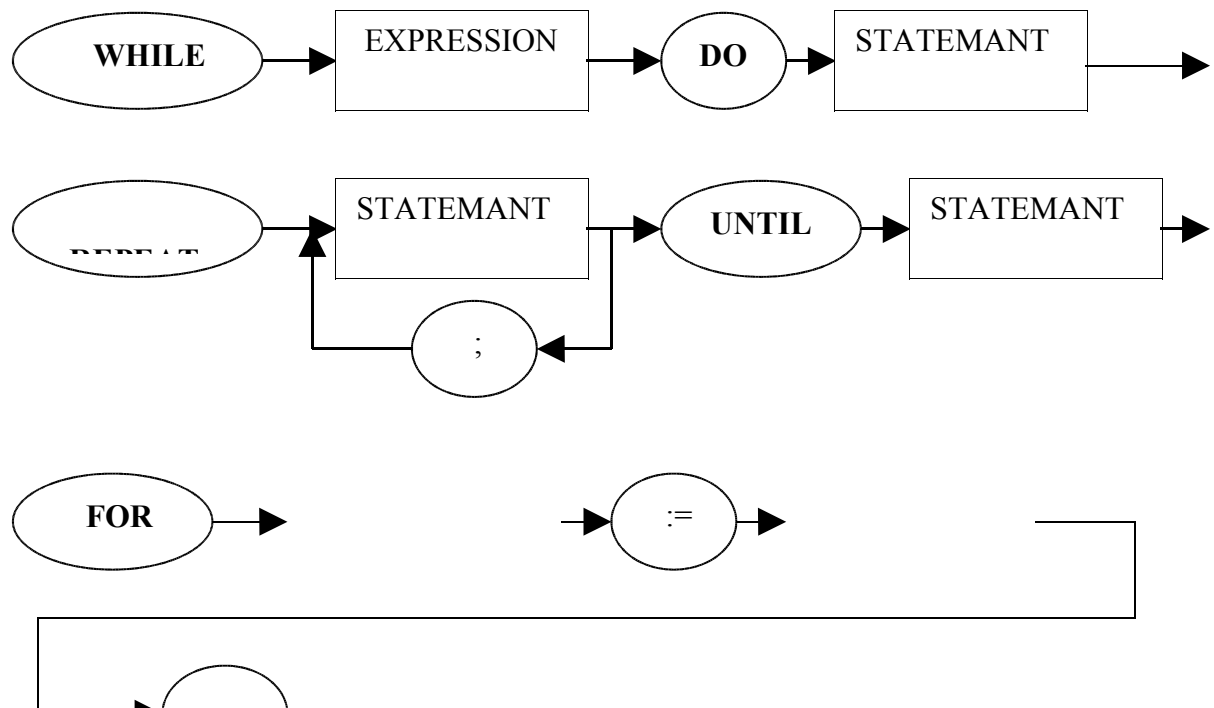
Constant Declaration:

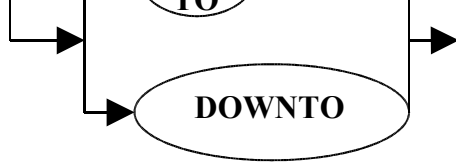


Esempio:

```
CONST min = 10;  
      max = 100;
```

3.8 ITERAZIONI





IDENTIFIER



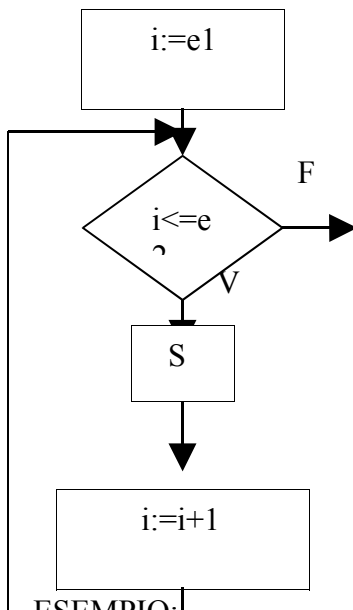
EXPRESSION

EXPRESSION

STATEMANT

ISTRUZIONE FOR:

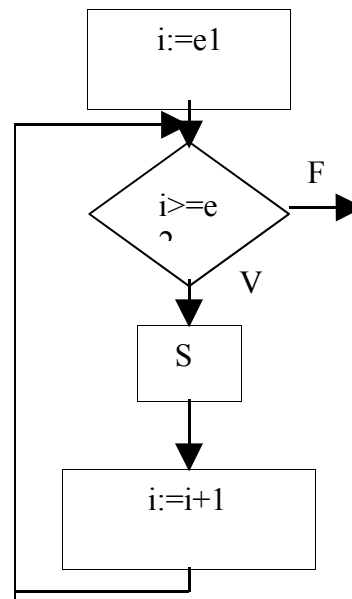
FOR i:=e1 TO e2 DO S



ESEMPIO:

CONST paganormale = 10000;
 Pagastraordinario = 20000;

FOR i:=e1 DOWNTWO e2 DO S



In generale:
 i:= succ(i)

TYPE giorno = (lun,mar,mer,gio,ven,sab,dom);

VAR paga : 0..maxint;
giornoferiale : lun..ven;
giornofestivo : sab..dom;
ore : 0..24;

BEGIN

Paga := 0;

FOR giornoferiale := lun **TO** ven **DO**

BEGIN

Read (ore);

paga := paga+ore*paganormale;

END;

FOR giornofestivo := sab **TO** dom **DO**

BEGIN

Read (ore);

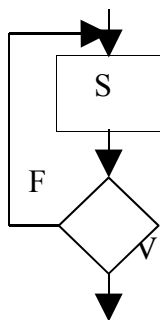
paga := paga+ore*orestraordinario;

END;

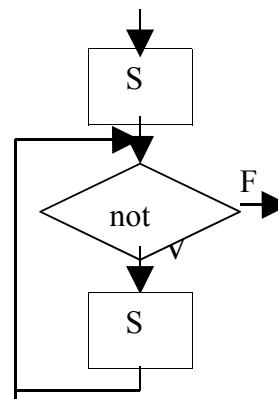
writeln ('paga=',paga);

END.

3.9 EQUIVALENZE



Equivalente a



REPEAT S UNTIL P

BEGIN

S;

WHILE not P **DO** S

END

