Il Progetto degli Algoritmi

(Il linguaggio degli schemi a blocchi: seconda parte)

Fondamenti di Informatica A

Ingegneria Gestionale

Università degli Studi di Brescia

Docente: Prof. Alfonso Gerevini

Esempio: Problema 2

Algoritmo per il calcolo del Massimo Comune Divisore (MCD) di due interi *x* ed *y*

Procedimento (descrizione <u>in linguaggio naturale</u>):

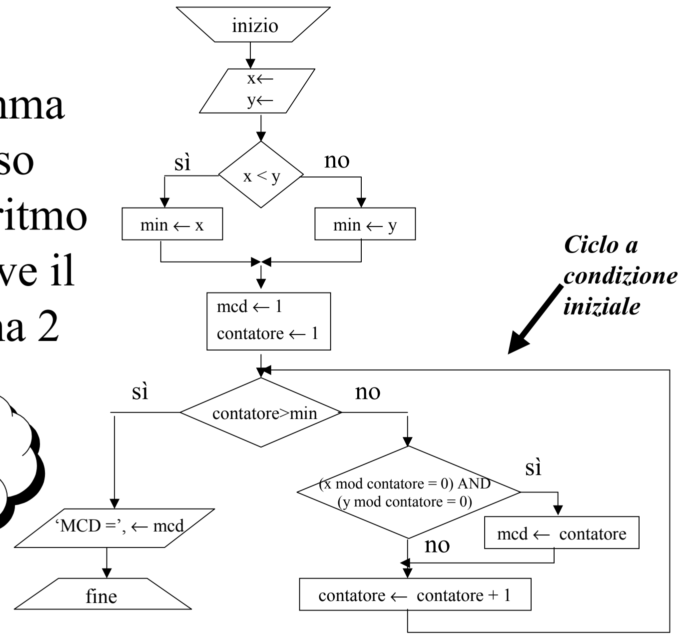
Fai la scansione di tutti i numeri compresi fra 1 e il <u>minimo</u> tra *x* ed *y*. Per ognuno stabilisci se è un divisore comune ad *x* ed *y* (come? *b* è divisore di *a* se *a* mod *b* = 0, ovvero se la divisione di *a* per *b* dà come resto 0). Ogni volta che un numero preso in considerazione risulta divisore sia di *x* che di *y* memorizzalo come attuale MCD in una variabile all'inizio posta a 1. Alla fine della scansione la variabile conterrà il valore desiderato.

Algoritmo per il Massimo Comune Divisore

Scomposizione in "sottoproblemi":

1. Leggi un valore dall'esterno e inseriscilo nella variabile *x*
2. Leggi un valore dall'esterno e inseriscilo nella variabile *y*
3. <u>Se</u> *x* < *y* <u>allora</u> *min* ← *x* <u>altrimenti</u> *min* ← *y*
4. *mcd* ← 1
5. *contatore* ← 1
6. <u>Se</u> *contatore* > min <u>allora</u> vai al passo 11 <u>altrimenti</u> vai al passo 7
7. <u>Se</u> ((*x* mod *contatore* = 0) e (*y* mod *contatore* = 0)) <u>allora</u> vai al passo 8 <u>altrimenti</u> vai al passo 9
8. *mcd* ← *contatore*
9. *contatore* ← *contatore* + 1
10. Torna al passo 6
11. Stampa "MCD =" seguito dal valore in *mcd*
12. Fine

Diagramma di flusso dell'algoritmo che risolve il problema 2



Ciclo a condizione iniziale

… e se *x* e *y* sono uguali?

Esecuzione passo passo dell'algoritmo

1 Lettura di due numeri e memorizzazione nelle variabili *x* e *y* (supponiamo di acquisire 6 e 9, quindi in memoria si avrà *x*=6 e *y*=9)
2 Controllo se x < y ➔ è vero
3 min ← 6
4 mcd ← 1, contatore ← 1
5 controllo se (contatore > min) ➔ non è vero
6 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **1** e 9 è divisibile per **1** ➔ è vero
7 mcd ← 1
8 contatore ← 2
9 controllo se (contatore > min) ➔ non è vero
10 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **2** e 9 è divisibile per **2** ➔ non è vero

Esecuzione passo passo dell'algoritmo (cont.)

11 contatore ← 3
12 controllo se (contatore > min) ➔ non è vero
13 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **3** e 9 è divisibile per **3** ➔ è vero
14 mcd ← 3
15 contatore ← 4
16 controllo se (contatore > min) ➔ non è vero
17 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **4** e 9 è divisibile per **4** ➔ non è vero
18 contatore ← 5
19 controllo se (contatore > min) ➔ non è vero
20 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **5** e 9 è divisibile per **5** ➔ non è vero

Esecuzione passo passo dell'algoritmo (cont.)

21 contatore ← 6
22 controllo se (contatore > min) ➔ non è vero
23 controllo se ((x mod contatore = 0) e (y mod contatore = 0)), cioè se 6 è divisibile per **6** e 9 è divisibile per **6** ➔ non è vero
24 contatore ← 7
25 controllo se (contatore > min) ➔ è vero
26 Stampa "MCD = 3" (cioè stampo il valore nella variabile *mcd*)

1-26 = <u>SEQUENZA DI COMPUTAZIONE</u>

Esempio: Problema 2

Algoritmo di Euclide per il calcolo del Massimo Comun Divisore (MCD) di due interi ➔ un altro algoritmo che risolve lo *stesso* problema

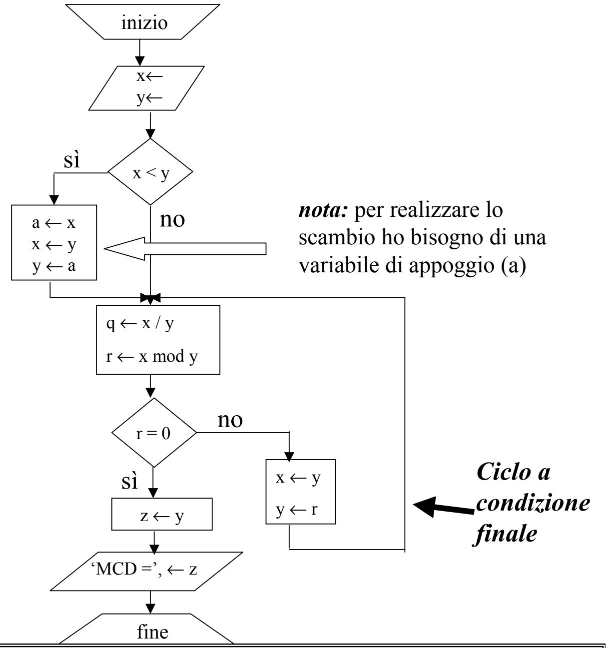
Scomposizione in sottoproblemi:

1. Leggi un valore dall'esterno e inseriscilo nella variabile *x*
2. Leggi un valore dall'esterno e inseriscilo nella variabile *y*
3. <u>Se</u> *x* < *y* <u>allora</u> scambia *x* con *y*
4. Dividi *x* per *y*: sia *q* il quoziente ed *r* il resto*
5. <u>Se</u> *r* = 0 <u>allora</u> vai al passo 8
6. *y* ←*x*, *y* ← *r*
7. <u>Vai</u> al passo 3
8. *z* ← *y*
9. Stampa/visualizza la frase "MCD = " seguita dal valore in *z*
10. Fine

… e i rami "altrimenti" ?

*operazioni: divisione intera e modulo

Diagramma di flusso dell’algoritmo di Euclide che risolve il problema 2



Esecuzione passo passo dell’algoritmo che risolve il problema 2

1 Lettura di due numeri e memorizzazione nelle variabili *x* e *y* (supponiamo di acquisire 6 e 9, quindi in memoria si avrà *x*=6 e *y*=9)
2 Controllo se x < y ➔ è vero
3 Faccio lo scambio: a ←6, x ←9, y ←6
4 divisione intera di x per y ➔ q ←1, resto della divisione ➔ r ← 3
5 controllo se r = 0 ➔ non è vero
6 x ← 6, y ← 3
7 divisione intera di x per y ➔ q ← 2, resto della divisione ➔ r ← 0
8 controllo se r = 0 ➔ è vero
9 z ← 3
10 Stampa/visualizza “MCD = 3”
11 Fine

Esercizio: Rifare l’esecuzione passo passo assumendo di acquisire ***x* = 18** e ***y* = 12**

Esempio: Problema 2 Algoritmo di Euclide

Si basa sulla ***constatazione*** che:

Se $x = y$ allora **MCD(*x*,*y*) = *x* (oppure *y*)**

Se $x \neq y$ allora, supponendo **x > y**, **MCD(*x*,*y*) = MCD(*x-y*,*y*)**

Infatti:

1. Se $x > y$ e k è un divisore comune a x e a y, allora k è anche un divisore di x-y. Infatti $x = k * d$ e $y = k * r$ per qualche intero positivo d e r. Quindi*:* x-$y = k*(d-r)$, essendo (d-r) ancora un intero positivo
2. Allo stesso modo è possibile dimostrare che se k è un divisore comune ad x-y e a y allora è un divisore anche di x
3. Quindi tutti i divisori comuni di x e y coincidono con i divisori comuni di x-y e y, dunque anche i massimi comuni divisori fra le due coppie di numeri coincidono

Quindi:

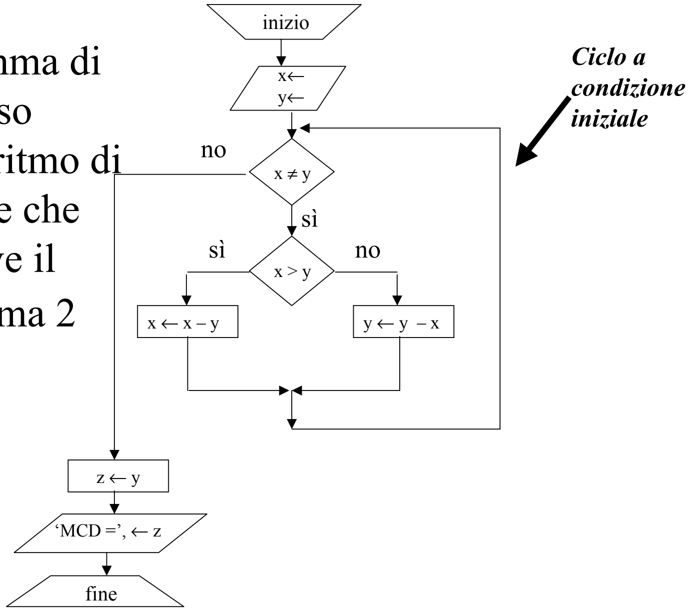
se chiamiamo $s = x$-y si può anche dire, se $s > y$, MCD(s,y) = MCD(s-y,y), altrimenti se $y > s$, MCD(y,s) = MCD(y-s,s)

Esempio: Problema 2 L’algoritmo di Euclide in forma più semplice

Scomposizione in sottoproblemi:

1. Leggi un valore dall’esterno e inseriscilo nella variabile x
2. Leggi un valore dall’esterno e inseriscilo nella variabile y
3. Se $x \neq y$ allora vai al passo 4 altrimenti vai al passo 6
4. Se $x > y$ allora $x \leftarrow x - y$ altrimenti $y = y - x$
5. Vai al passo 3
6. $z \leftarrow y$
7. Stampa/visualizza la frase “MCD = ” seguita dal valore in z
8. Fine

Diagramma di flusso dell'algoritmo di Euclide che risolve il problema 2



Esecuzione passo passo dell'algoritmo

1 Lettura di due numeri e memorizzazione nelle variabili *x* e *y* (supponiamo di acquisire 6 e 9, quindi in memoria si avrà *x*=6 e *y*=9)
2 Controllo se x ≠ y ➔ è vero
3 Controllo se x > y ➔ non è vero
4 y ← 3 (infatti: y ← 9 – 6)
5 Controllo se x > y ➔ non è vero
6 Controllo se x > y ➔ è vero
7 x ← 3 (infatti: x ← 6 – 3)
8 Controllo se x ≠ y ➔ non è vero
9 z ← 3
10 Stampa/visualizza "MCD = 3"
11 Fine

Esercizio: Rifare l'esecuzione passo passo assumendo di acquisire ***x* = 24** e ***y* = 9**

Esempio: Problema 3

Determinare il massimo fra *n* numeri positivi, finchè non si inserisce 0

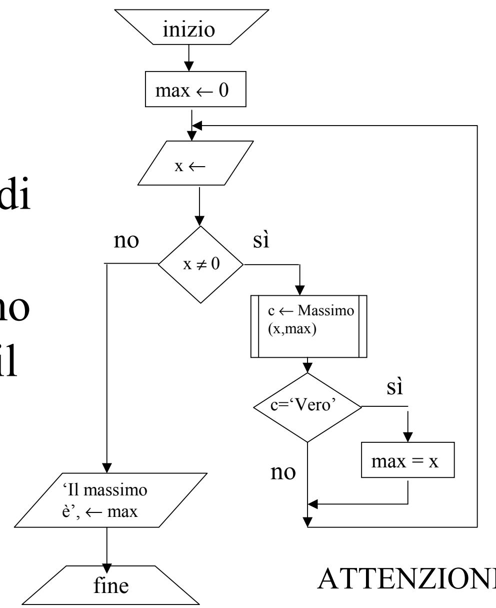
Scomposizione in sottoproblemi:

1 *max* ← 0 → *nota:* la variabile *max* conterrà il risultato finale
2 Leggi un valore dall'esterno e inseriscilo nella variabile *x*
3 Se *x* ≠ 0 allora vai al passo 4 altrimenti vai al passo 7
4 Chiama il sottoprogramma *Massimo(x, max)* e inserisci il risultato nella variabile *c*
5 Se *c* è uguale a 'Vero' allora *max=x* → *nota: 'c'* è una variabile "logica"
6 Torna al passo 2
7 Stampa la frase "Il massimo è" seguita dal valore contenuto in *max*
8 Termina l'esecuzione

ciclo

NOTA: *Massimo(x,max)* è un sottoprogramma che descrive la soluzione del seguente sottoproblema: ricevendo in ingresso *x* e *max*, ritorna un valore Vero se *x>=max* e un valore Falso se *x<max* (leggermente diverso dal problema 1… **esercizio:** fare lo schema a blocchi)

Diagramma di flusso dell'algoritmo che risolve il problema 3



ATTENZIONE!!!!

Attenzione!

- Anche per problemi limitatamente complessi lo schema a blocchi può diventare intricato
- Al crescere del numero degli archi e dei blocchi lo schema si fa complesso
- I diagrammi diventano NON strutturati
- Il diagramma precedente è un esempio di "*diagramma **non** strutturato*"

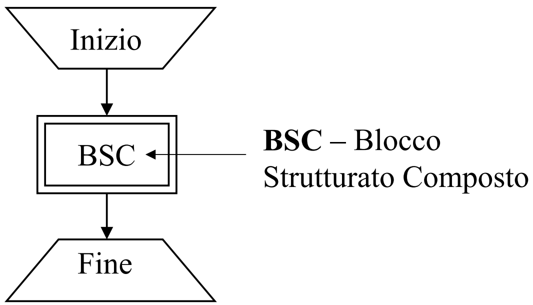
Problema Linguaggio SB

- Il linguaggio visto è da "troppa libertà" e consente di creare SB di cattiva qualità (ad es. difficile lettura e modifica)
- Occorrono regole più rigide per disciplinare la costruzione di uno SB (ed avere garanzia che il risultato sia buono)

⇒ Restringiamo il linguaggio ponendo dei vincoli

Diagrammi strutturati

- Esistono dei <u>vincoli</u> da rispettare per progettare buoni programmi



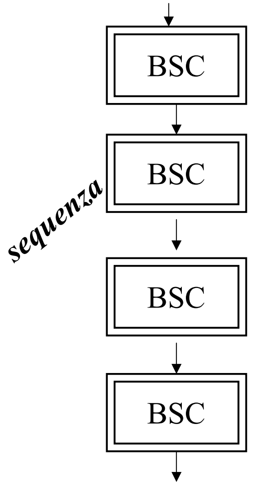
Tipi di Blocchi Strutturati Composti

- Sequenza
- Selezione a 2 vie (struttura condizionale doppia)
- Ciclo a condizione iniziale

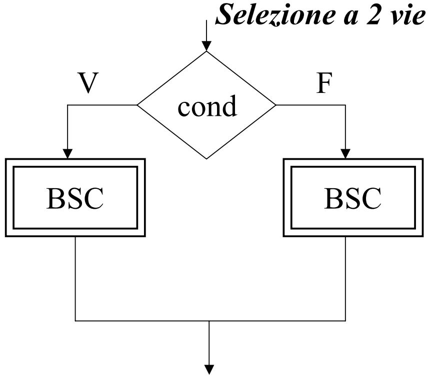
Strutture di controllo

- Blocco funzionale semplice

BSC – Blocco Strutturato Composto

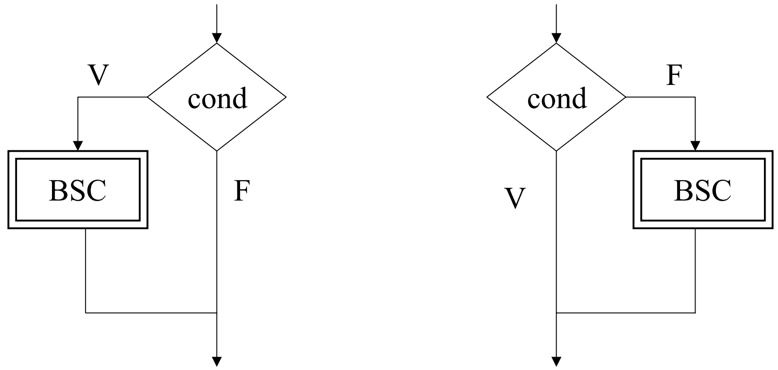


Selezione a 2 vie



BSC – Blocco Strutturato Composto

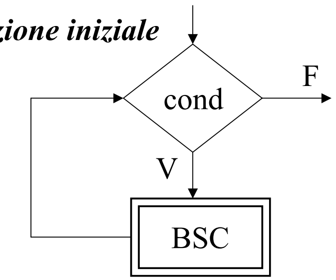
Selezione semplice



Si usa un *blocco funzionale nullo*

BSC – Blocco Strutturato Composto

Ciclo a condizione iniziale



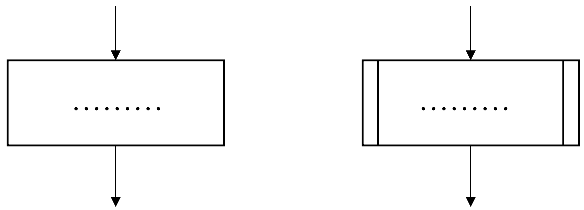
Si resta nel ciclo finchè la condizione rimane vera

La posizione di V ed F non è indifferente

Di solito si usa questa convenzione, in modo tale che **'cond'** abbia il significato di ***condizione di ingresso nel ciclo***

BSC – Blocco Strutturato Composto

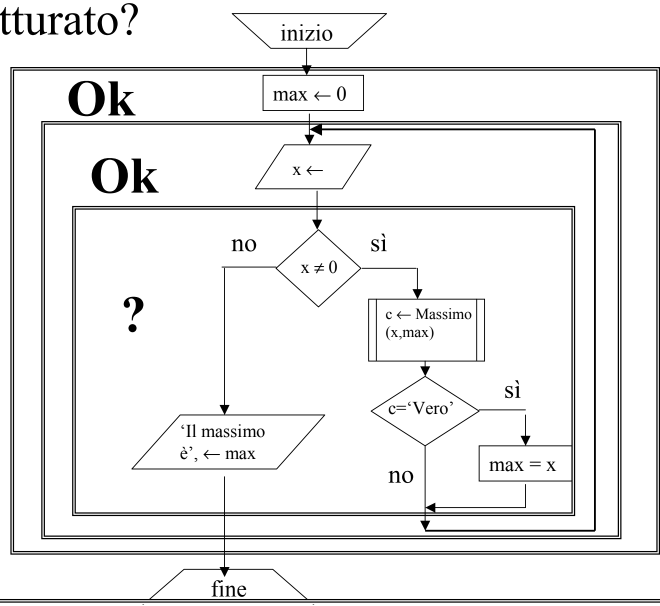
Blocco di elaborazione semplice



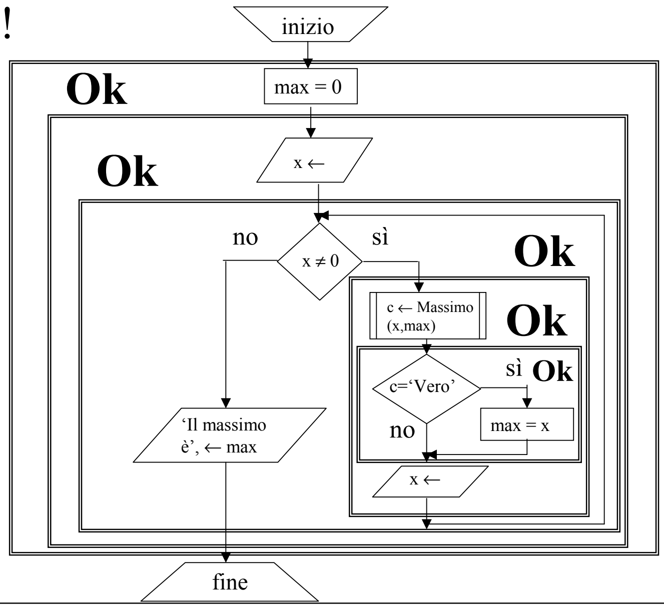
Sviluppo di uno schema a blocchi strutturato

Dato lo schema iniziale costituito da blocchi ***Inizio***, ***BSC*** e ***Fine*** si sostituisce passo passo ad ogni ***BSC*** una delle *strutture di controllo* (sequenza, selezione, ciclo) oppure un *blocco funzionale semplice* fino a quando tutti i blocchi strutturati composti sono stati eliminati

E' strutturato?

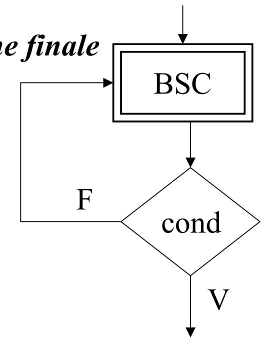


Ora sì!



Questo non è un BSC !!!

Ciclo a condizione finale



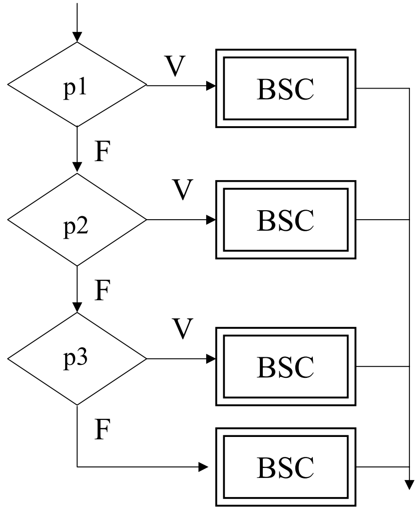
Si resta nel ciclo fino a quando la condizione diventa vera

E' comunque una struttura di controllo molto utilizzata e quindi la considereremo come se fosse BSC

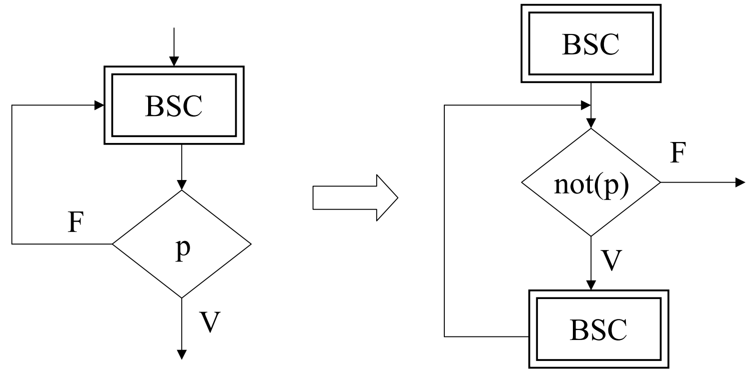
Questo non è un BSC !!!

Struttura condizionale multipla

E' anche questa una struttura di controllo molto utilizzata che quindi considereremo come se fosse un BSC

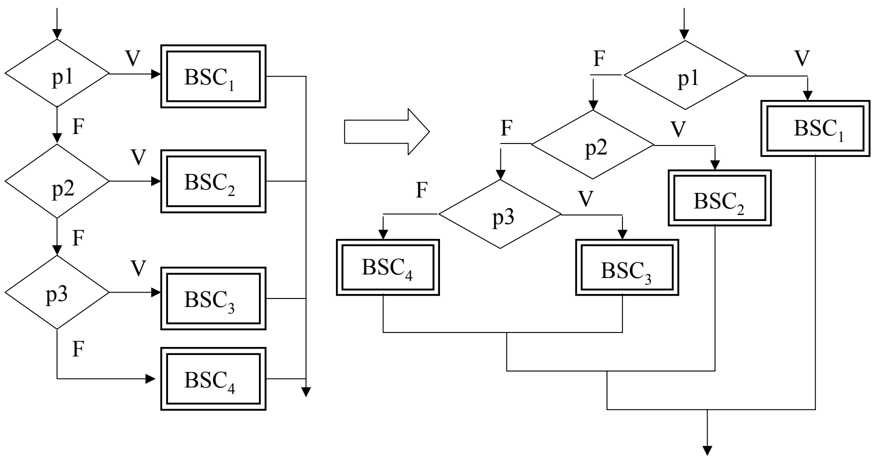


Trasformazione funzionale (1)

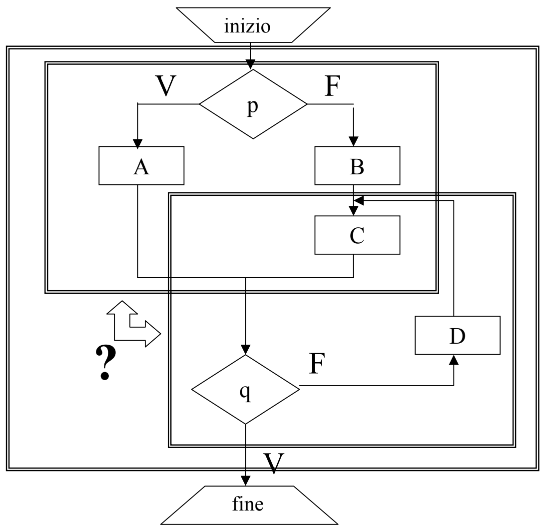


<u>Infatti</u>: la potenza espressiva non cambia, guadagno in termini di notazione

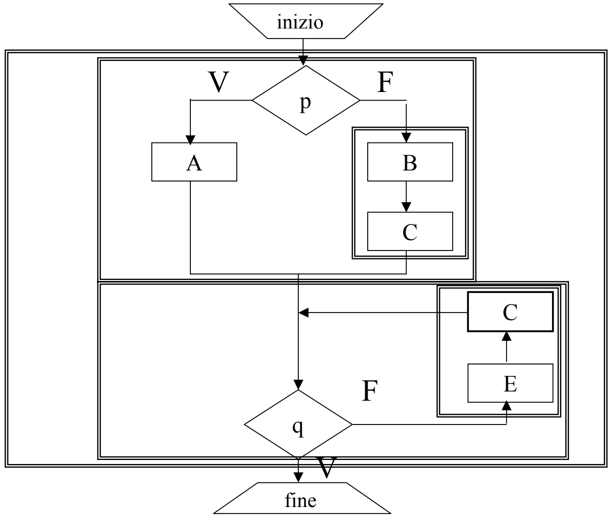
Trasformazione funzionale (2)



Altro esempio: è strutturato?



Trasformazione algoritmica (richiede al più la duplicazione di blocchi)



Equivalenze tra 2 SB X e Y

Equivalenza Debole (o funzionale): se e solo se, per ogni possibile dato di ingresso i, dette Fx e Fy le *funzioni calcolate* da X e Y rispettivamente, o Fx(i) e Fy(i) sono entrambe indefinite o Fx(i) = Fy(i)

Equivalenza forte (o algoritmica): se e solo se, per ogni possibile dato di ingresso i, dette Sx e Sy le *sequenze di computazione* generate da X e Y rispettivamente, Sx(i) = Sy(i)

Se X è equivalente fortemente a Y, allora X è debolmente equivalente a Y

SB generici e SBS

Gli SB non strutturati sono più "potenti" degli SBS?

Dal punto di vista funzionale NO!

Teorema: per ogni SB X non strutturato esiste uno SBS Y debolmente equivalente a X (ottenibile attraverso *trasformazione funzionale*)

La trasformazione funzionale richiede in generale l'introduzione di nuovi blocchi funzionali semplici e nuovi blocchi decisionali.