Fondamenti di Informatica 1

Alfonso Miola

Problemi, Algoritmi e Programmi

Dispensa 2

Gennaio 2001

Contenuti

- Cosa è un problema
- Esempio di problema: il MCD
- Cosa è un algoritmo
- L'algoritmo di Euclide
- Problemi decidibili e non
- Esempio del prodotto di interi
- Correttezza ed efficienza
- Programmazione strutturata

Premessa

- **Vengono introdotti alcuni concetti base dell'Informatica: Problema, Algoritmo, Programma, Processo**
- **In modo sintetico, attraverso semplici esempi, sono presentati i contenuti principali del corso che verranno poi ripresi in maniera più estesa**
- **In particolare ci si concentra su aspetti di correttezza e di efficienza della soluzione automatica di problemi, anche attraverso l'uso di metodi di programmazione strutturata**

Problema

- **Problemi di interesse sono quelli per cui è possibile una precisa formalizzazione, in un qualche formalismo espressivo, ad esempio quello offerto dalla matematica.**
- **Esistono problemi che sono tanto ben esprimibili e comprensibili da poterne definire una strategia di soluzione basata sull'applicazione sistematica di ben precise regole operative che consente di ottenere i risultati attesi a partire dai dati disponibili.**
- **In molti di questi casi è possibile affidare l'applicazione delle regole di soluzione ad un esecutore specializzato in grado di svolgere (più o meno) rapidamente i compiti affidatigli.**

Un esempio

“Determinare il Massimo Comun Divisore di due numeri interi positivi”

Enunciato informale del problema in linguaggio naturale

Un esempio . . .

Determinare $z = MCD(x, y)$

dove $x, y, z \in N^+$, $x \geq y$

Formalizzazione del problema in linguaggio naturale e matematico, in cui compaiono i simboli x, y, e z da interpretare come oggetti appartenenti al dominio N^+ dei numeri naturali positivi e il simbolo MCD da interpretare come la funzione matematica Massimo Comun Divisore

Massimo Comun Divisore

$$MCD(x, y) = \max(D(x) \cap D(y))$$

$$D(x) = \{ d \in N^+ \mid \exists\ q \in N, x = d \cdot q \}$$

$$D(y) = \{ d \in N^+ \mid \exists\ q \in N, y = d \cdot q \}$$

Definizione matematica del MCD

Massimo Comun Divisore . . .

Soluzione 1

Applico la definizione matematica:

- **calcolo i due insiemi Dx e Dy dei divisori di x e di y, rispettivamente**
- **costruisco l'insieme intersezione di Dx e Dy**
- **determino il valore massimo dell'insieme intersezione**

Oppure . . .

Soluzione 2

- **Individuo il valore m minimo tra x e y**
- **Verifico se m divide sia x sia y; se sì allora m è il MCD(x,y), e ho finito**
- **Decremento m di 1 e ripeto il passo precedente, al più fino al valore m = 1**

Devo quindi eseguire un numero di divisioni al più pari a 2m

Oppure . . .

Soluzione 3

Cerco altre proprietà del MCD . . .

Siano $x, y, r \in N^+$, $q \in N$.

Se $x = q \cdot y + r$ allora

$D(x) \cap D(y) = D(y) \cap D(r)$

Teorema (Proprietà) del MCD

Massimo Comun Divisore ...

Sulla base di questa proprietà si può definire la sequenza di regole da applicare per ottenere il MCD di due qualsiasi numeri naturali positivi x e y:

- **r = Mod (x , y)** **(Mod è la divisione con resto)**
- **se r = 0 allora y è il MCD(x, y), altrimenti il problema si riconduce al più semplice problema che è quello di ottenere MCD(y, r)**

Massimo Comun Divisore ...

Tale sequenza di regole rappresenta un metodo risolutivo del problema posto, detto Algoritmo, nel caso specifico questo è noto come Algoritmo di Euclide (AE)

Ad esempio il calcolo di MCD (187,34) = 17 procede così:

187 = 5 34 + 17 **(17 = Mod (187,34))**

34 = 2 17 + 0 **(0 = Mod (34,17))**

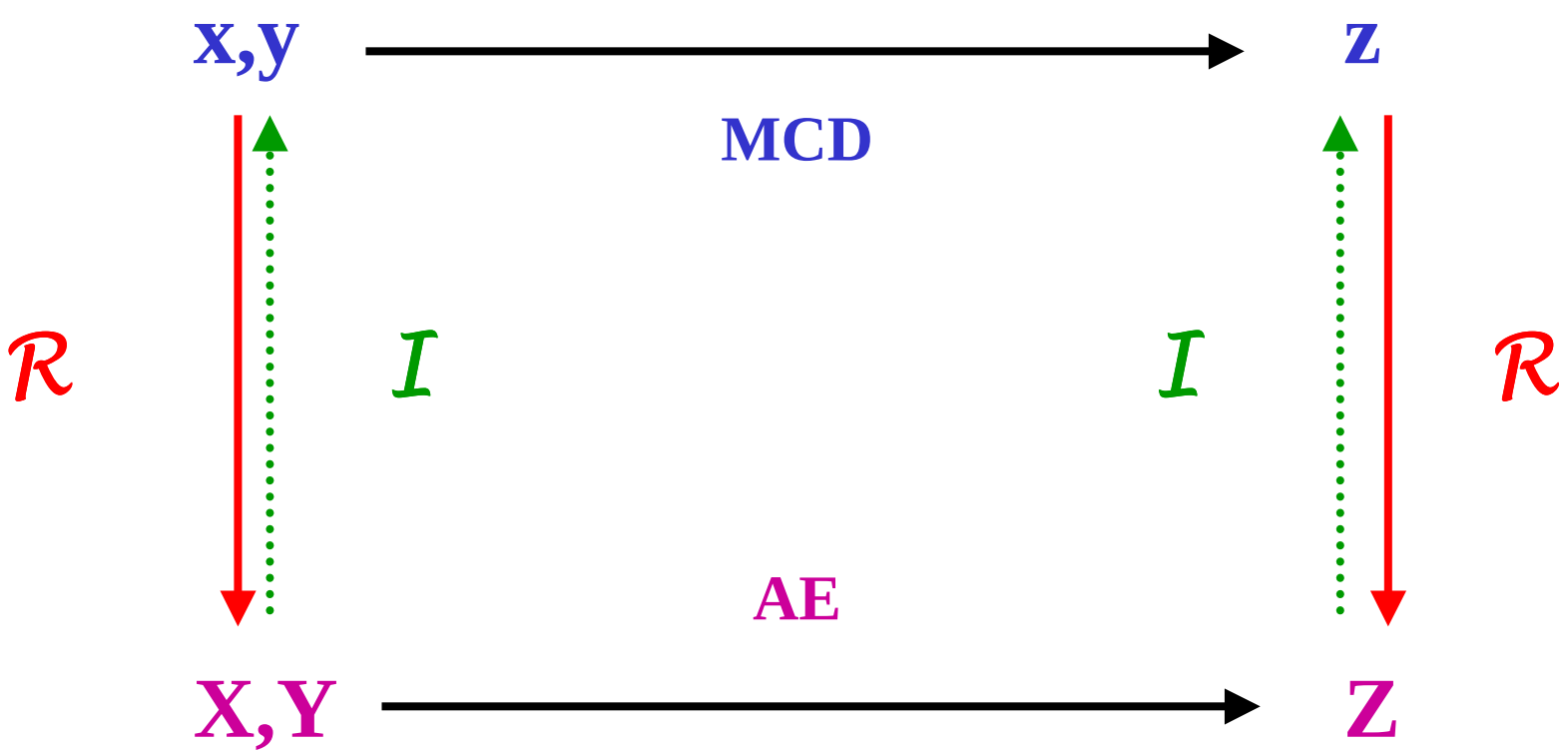
Rappresentazione e interpretazione

Si noti che le regole dell'algoritmo AE sono in effetti applicate su possibili rappresentazioni X, Y, e Z dei numeri interi x, y e z (oggetti astratti) che sono le loro interpretazioni rispettive

Una rappresentazione è ovviamente quella nel sistema di numerazione arabo decimale, come nell'esempio

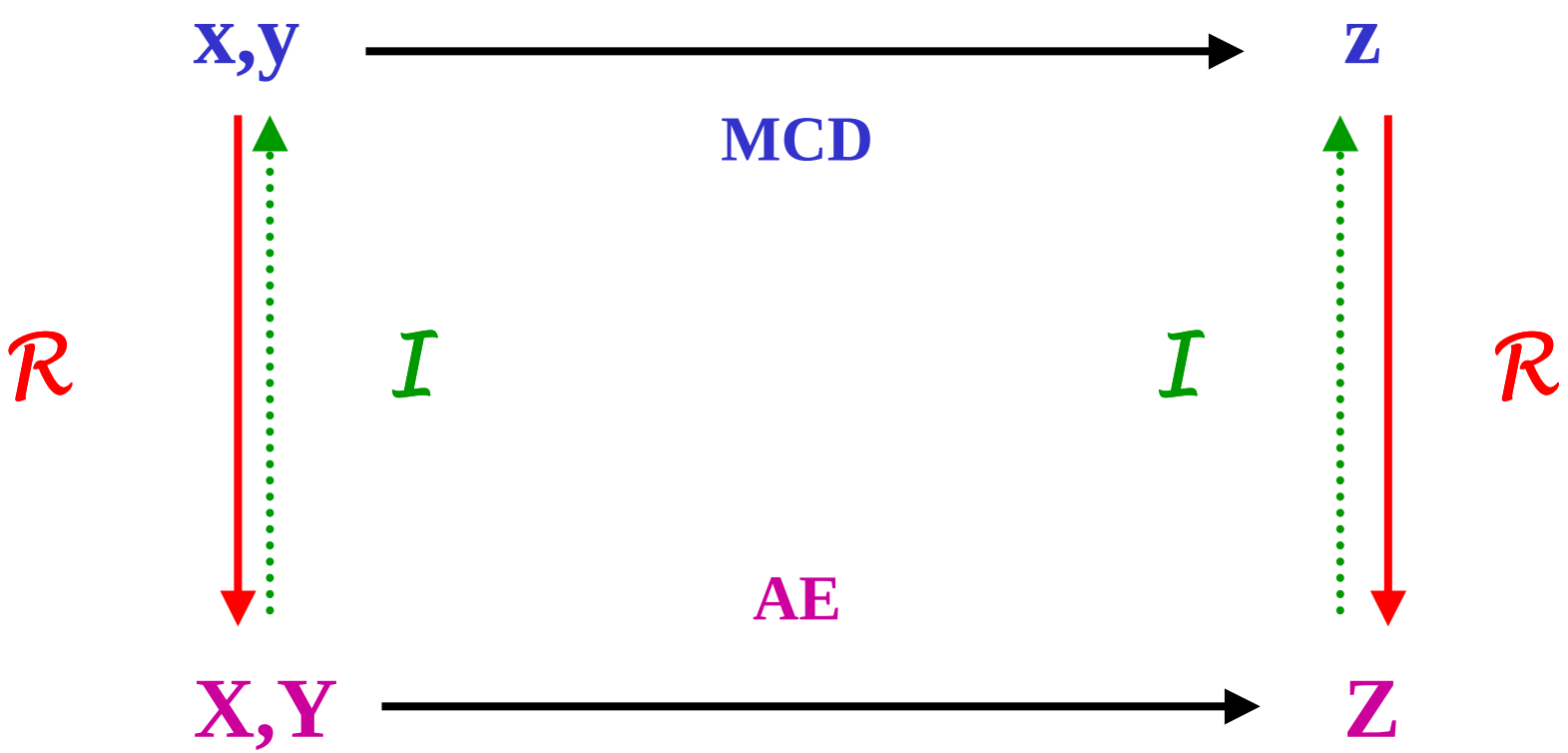
Cioè . . .

$$X = \mathcal{R}(x)\ ,\ Y = \mathcal{R}(y)\ ,\ Z = \mathcal{R}(z)$$
$$x = \mathcal{I}(X)\ ,\ y = \mathcal{I}(Y)\ ,\ z = \mathcal{I}(Z)$$



Cioè ancora . . .

$$AE(\mathcal{R}(x), \mathcal{R}(y)) = \mathcal{R}(MCD(x, y))$$
$$\mathcal{I}(AE(X, Y)) = MCD(\mathcal{I}(X), \mathcal{I}(Y))$$



Algoritmo

Procedimento risolutivo di un problema

Insieme di regole che, eseguite ordinatamente, permettono di ottenere i risultati del problema a partire dai dati a disposizione

Perché un insieme di regole possa considerarsi un algoritmo deve rispettare alcune proprietà . . .

Proprietà di un algoritmo

Non ambiguità - Le istruzioni devono essere univocamente interpretabili dall'esecutore dell'algoritmo (nel seguito l'esecutore sarà l'elaboratore, ma il problema si pone anche per esecutori umani)

Eseguibilità - L'esecutore deve essere in grado, con le risorse a disposizione, di eseguire ogni istruzione, ed in tempo finito

Finitezza - L'esecuzione dell'algoritmo deve terminare in un tempo finito per ogni insieme di valori di ingresso

Problemi decidibili

- **Dati 2 numeri, calcolarne la somma**
- **Dati n numeri, calcolarne la somma**
- **Dato un elenco nomi/numeri telefonico, trovare il numero telefonico di una data persona**
- **Consultare una carta geografica**
- **Progettare una rete elettrica / un ponte**

Problemi non decidibili

- **Decidere se f(x) è una funzione costante**
- **Determinare il cambio Euro / Dollaro al 1/1/2020**

Efficienza temporale

Problema:

"Ricerca il valore massimo in un insieme numerico"

$$\{ a_1, a_2, ..., a_n \}$$

Definizione matematica:

Esiste un indice i tale che $a_i \geq a_j$ *, per* $j = 1,2,.....,n$

Efficienza temporale . . .

Soluzione 1

- Applico la definizione matematica e verifico la proprietà per ogni valore di **i** da **1** a **n**
- Devo quindi eseguire un numero di confronti pari a

$$(n-1)^2 = O(n^2)$$

cioè dell'**ordine di n^2**

Efficienza temporale . . .

Soluzione 2

- Assumo a_1 come massimo (provvisorio)
- Confronto con il successivo e decido il nuovo massimo (provvisorio); e così via di seguito . . .
- Devo quindi eseguire un numero di confronti pari a

$$n - 1 = O(n)$$

cioè dell'ordine di n

Efficienza spaziale

Supponiamo di avere una matrice di n x n numeri interi, di cui solo k sono non nulli,

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Vogliamo memorizzarla in modo efficiente

Efficienza spaziale . . .

Soluzione 1

Memorizziamo tutti gli elementi, uno dopo l'altro, riga dopo riga, occupando così n^2 celle di memoria

Soluzione 2

Memorizziamo le dimensioni della matrice e il valore dominante; quindi memorizziamo solo i valori dei k elementi non nulli con le rispettive posizioni di riga e colonna, occupando quindi $3(k + 1)$ celle di memoria

Soluzioni effettive

ATTENZIONE !!!

Esistono dei problemi che, pur rientrando nel mondo del decidibile, per i quali quindi è possibile individuare un algoritmo risolutivo, ammettono soluzioni non effettive cioè con un costo temporale e/o spaziale insostenibile in pratica

Un esempio

Supponiamo di dover assemblare in orbita un apparato costituito da n parti componenti $a_1, a_2, \ldots\ldots\ldots\ldots, a_n$ di peso rispettivo $p_1, p_2, \ldots\ldots\ldots\ldots, p_n$, potendo usufruire di missili con portata massima P

Supponendo che $p_i < P$ e che $\Sigma p_i >> P$, si tratta di minimizzare il numero di missioni da compiere

Prendiamo il caso n = 50, dobbiamo verificare il peso totale trasportabile in una missione sommando i pesi delle parti componenti a due a due, a tre a tre, e così via

Compiamo quindi 2^{50} operazioni di confronto con P e se ciascuna costa un **ms** il tempo totale sarà di **40.000 anni**

COSTO ESPONENZIALE

Riassumendo

Per delegare ad un esecutore (automatico) la soluzione di un problema, a partire dalla struttura astratta di appartenenza degli oggetti da trattare (manipolare), si deve perciò:

- **individuare una rappresentazione della struttura: degli oggetti e delle operazioni**
- **individuare una descrizione dell'algoritmo, come sequenza di passi elementari**

Rappresentazione

Per tali rappresentazioni si fa uso di un opportuno linguaggio

Nel caso di un esecutore automatico (elaboratore) si fa uso di un adeguato linguaggio di programmazione composto di regole comprensibili sia per noi che per l'elaboratore

Definizioni

- **dati di ingresso (INPUT)** - le rappresentazioni (fornite all'elaboratore) delle informazioni a disposizione
- **programma** - la rappresentazione dell'algoritmo nel linguaggio di programmazione scelto (un programma sarà costituito da un insieme di istruzioni)
- **dati di uscita (OUTPUT)** - le rappresentazioni fornite dall'elaboratore al termine della esecuzione del programma

PROBLEMA → **ALGORITMO**

ALGORITMO → **PROGRAMMA**

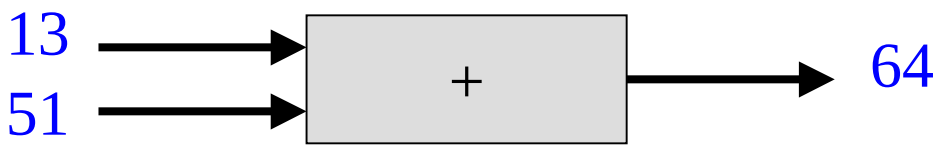
Elaboratore

- L'elaboratore è un **esecutore** di operazioni in sequenza che trasformano input in output
- Funziona secondo regole precise e conosce un numero (molto) limitato di comandi
- Esegue ciascun comando in modo univoco e in tempo finito
- Esegue **azioni** elementari (in numero limitato) su **dati** (che sono rappresentazioni di oggetti astratti) eseguendo **istruzioni**
- Le **azioni** procurano **effetti** di cambiamento di stato

Programma e processo

- **Un programma è una sequenza di istruzioni**
- **Un processo (di esecuzione di un programma) è una sequenza di azioni (che sono l'esecuzione delle istruzioni del programma)**
- **Un processo fa passare l'elaboratore da uno stato iniziale ad uno stato finale**

Sommatore



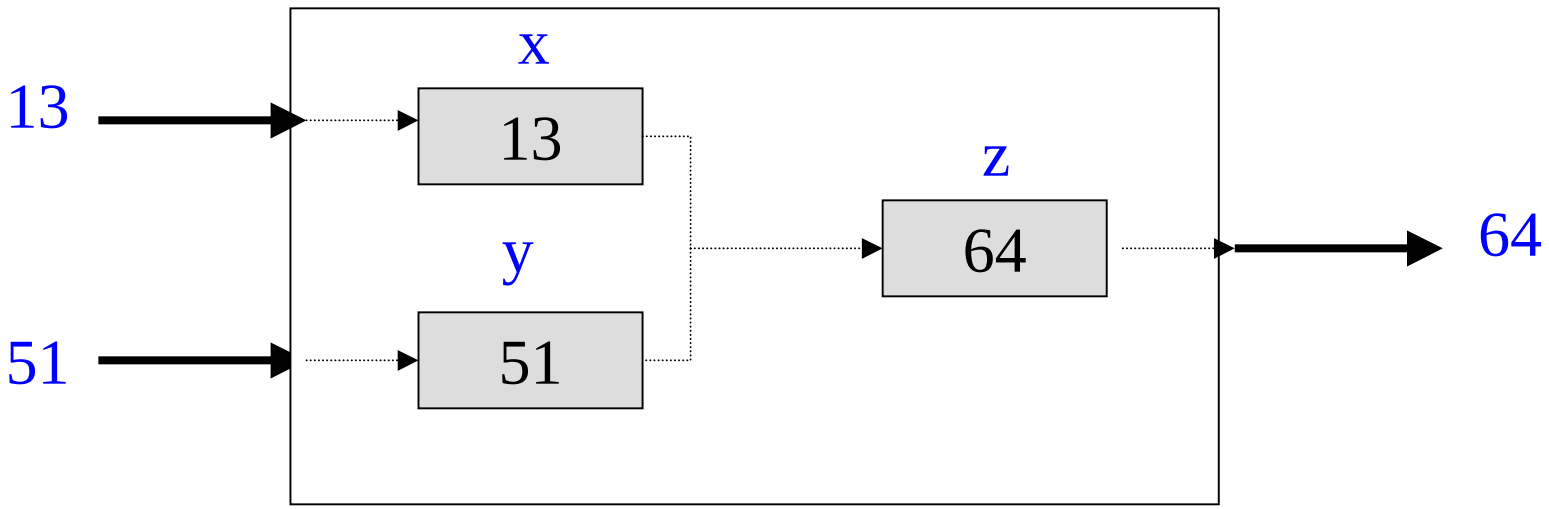
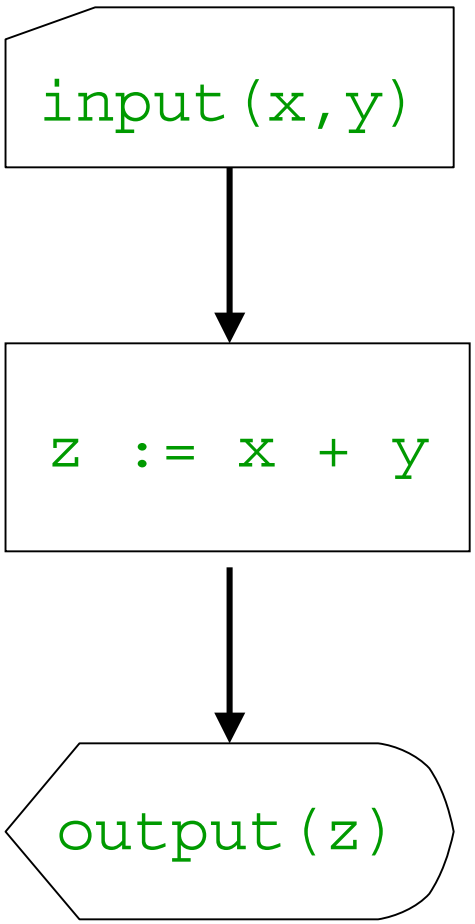


Diagramma a blocchi



Cosa è una variabile

Dal punto di vista logico una variabile è un simbolo che identifica la rappresentazione di un oggetto di interesse (di tipo fissato)

Dal punto di vista fisico una variabile può essere intesa come un contenitore per valori di un oggetto di interesse (di tipo fissato)

Alle variabili assegno quindi valori costanti

alfa = 137

oppure i valori di un'altra variabile

beta = alfa

Un altro problema

Determinare il prodotto di due numeri interi positivi

Enunciato informale in linguaggio naturale

Determinare $Z = X \bullet Y$, dove $X, Y, Z \in N^+$

Formalizzazione in linguaggio naturale e matematico

Un altro problema . . .

Se l'esecutore scelto per la soluzione del problema opera sulla seguente rappresentazione della struttura matematica degli oggetti da trattare:

$$\mathcal{NAT} = < N_{10} , + , - , * , < , = , 0 , 1 >$$

allora il problema e' risolto dall'esecuzione della seguente istruzione

```
z = x * y
```

x = Rapp (X) , y = Rapp (Y) , z = Rapp (Z)

Un altro problema . . .

Qualora invece l'esecutore scelto per la soluzione del problema operi su una diversa (meno espressiva e potente) rappresentazione della struttura degli oggetti da trattare, quale:

$$\mathcal{NAT} = < \mathrm{N}_{10} , + , - , < , = , 0 , 1 >$$

allora e' necessario definire un algoritmo risolutivo basato sull'applicazione di una opportuna sequenza di operazioni elementari tra quelle possibili (eseguibili dall'esecutore)

Un altro problema . . .

Teorema:

Siano x, y $\in$ N^+

Se y = 1 allora x • y = x

Altrimenti x • y = x + (x • (y - 1))

Cioè per avere il valore z del prodotto di x e y posso sommare x a sé stesso tante volte quanto il valore di y

In pratica . . .

Inizialmente il valore di z è 0, quindi, verificando passo dopo passo che il valore di y resti diverso da 0, ripeto queste due operazioni

- **incremento il valore corrente di z con x**
- **decremento di 1 il valore di y**

Algoritmo per il prodotto

1. z = 0

2. mentre y > 0

2.1. incrementa z di x

2.2. decrementa y di 1

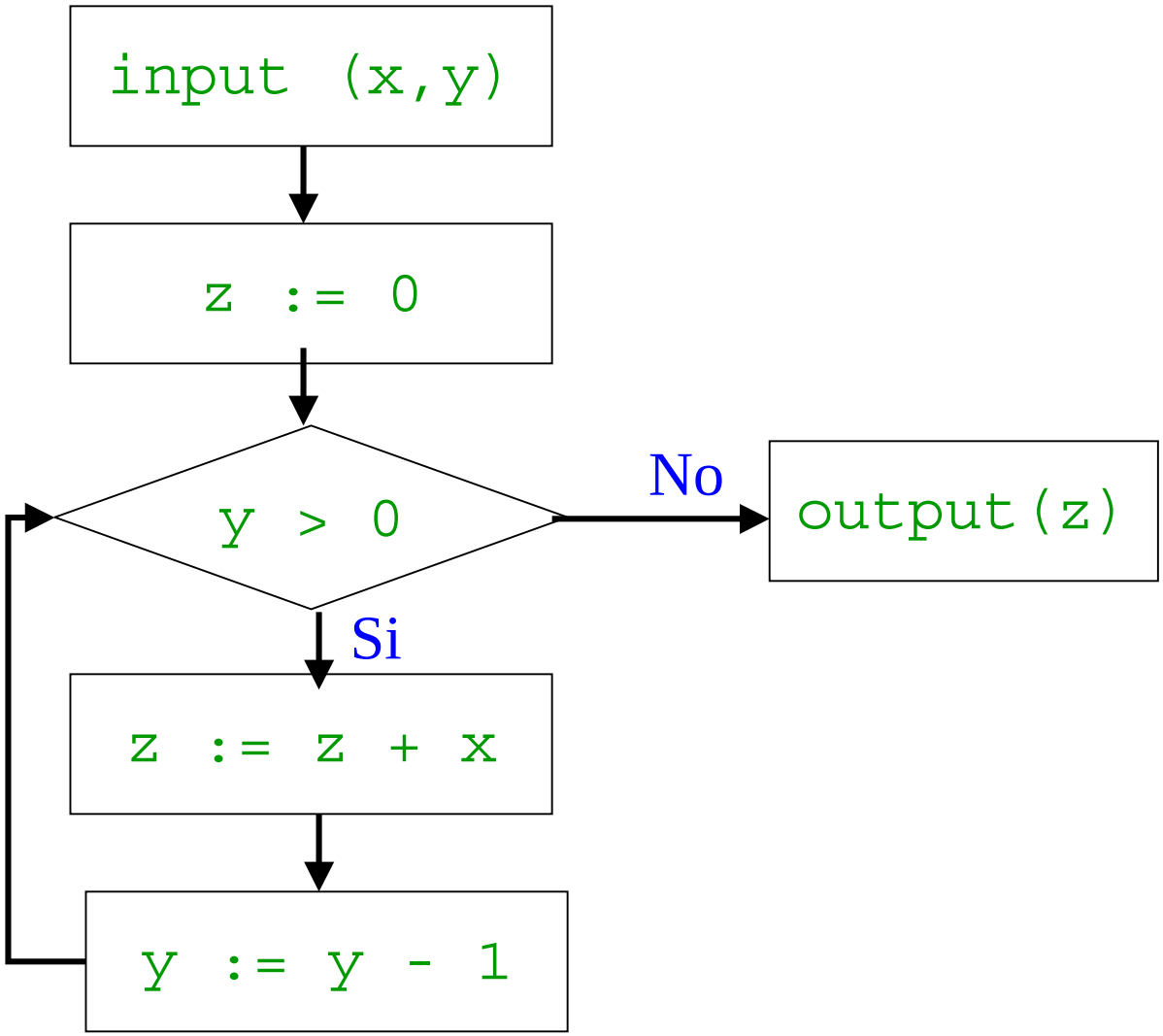
Descrizione dell'algoritmo

Pseudo-programma per il prodotto

```
Programma:      Prodotto
Variabili:      integer x, y;
Istruzioni:     input (x, y);
                z := 0;
                while (y > 0) {
                      z := z + x;
                      y := y - 1;
                }
                output (z);
```

Descrizione dell'algoritmo in un linguaggio (di programmazione)

Diagramma a blocchi



Traccia del programma

	X	Y	Z
1.	17	3	
2.	17	3	0
3.	17	3	0
4.	17	3	17
5.	17	2	17
3.	17	2	17
4.	17	2	34
5.	17	1	34
3.	17	1	34
4.	17	1	51
5.	17	0	51
3.	17	0	51
6.	17	0	51

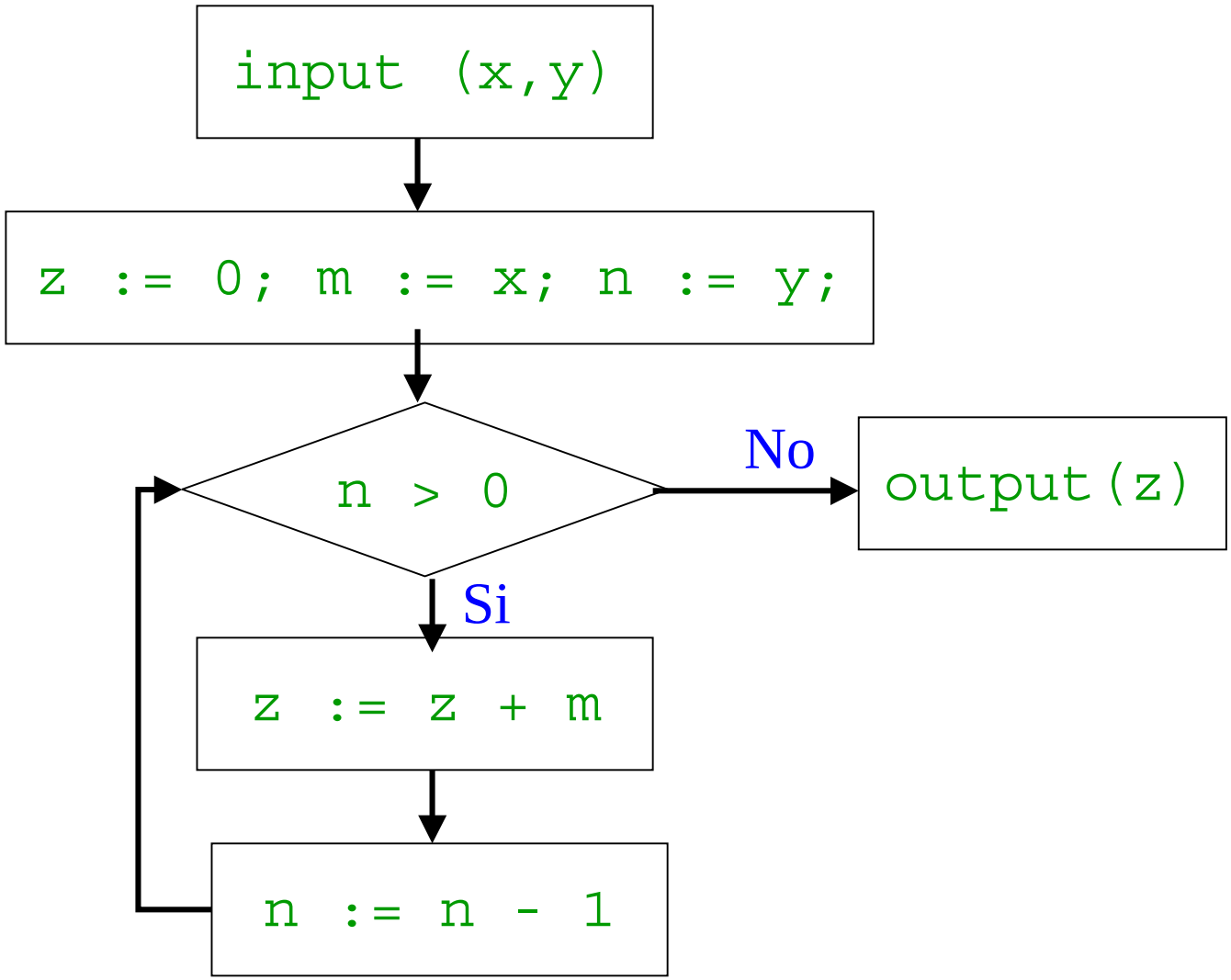
Una prima riflessione

Si dovrebbe poter verificare se il risultato è corretto, cioè se il valore finale di z sia o meno effettivamente il prodotto dei valori di x e di y

Al termine dell'elaborazione tuttavia il valore di y è nullo, si è cioè perduto il suo valore iniziale, determinando una impossibilità di verifica del risultato

Si può ovviare a questa carenza !!!

Diagramma a blocchi



Correttezza

```
Prodotto
        integer x, y, m, n;
        input (x, y);
        m := x;
        n := y;
        z := 0;
        while (n > 0) {
                z := z + m;
                n := n - 1:
        }
        output (z);
```

Correttezza . . .

	X	Y	M	N	Z	
1.	17	3	17	3		
2.	17	3	17	3	0	x*y = z+m*n
3.	17	3	17	3	0	
4.	17	3	17	3	17	
5.	17	3	17	2	17	x*y = z+m*n
3.	17	3	17	2	17	
4.	17	3	17	2	34	
5.	17	3	17	1	34	x*y = z+m*n
3.	17	3	17	1	34	
4.	17	3	17	1	51	
5.	17	3	17	0	51	x*y = z+m*n
3.	17	3	17	0	51	
6.	17	3	17	0	51	x*y = z+m*n

Un'altra riflessione

In questo caso il problema della correttezza può essere gestito tenendo anche conto della possibilità di determinare la validità della seguente proprietà durante l'esecuzione del programma

$$x * y = z + m * n$$

Questo tipo di proprietà è detto

invariante di ciclo

Efficienza

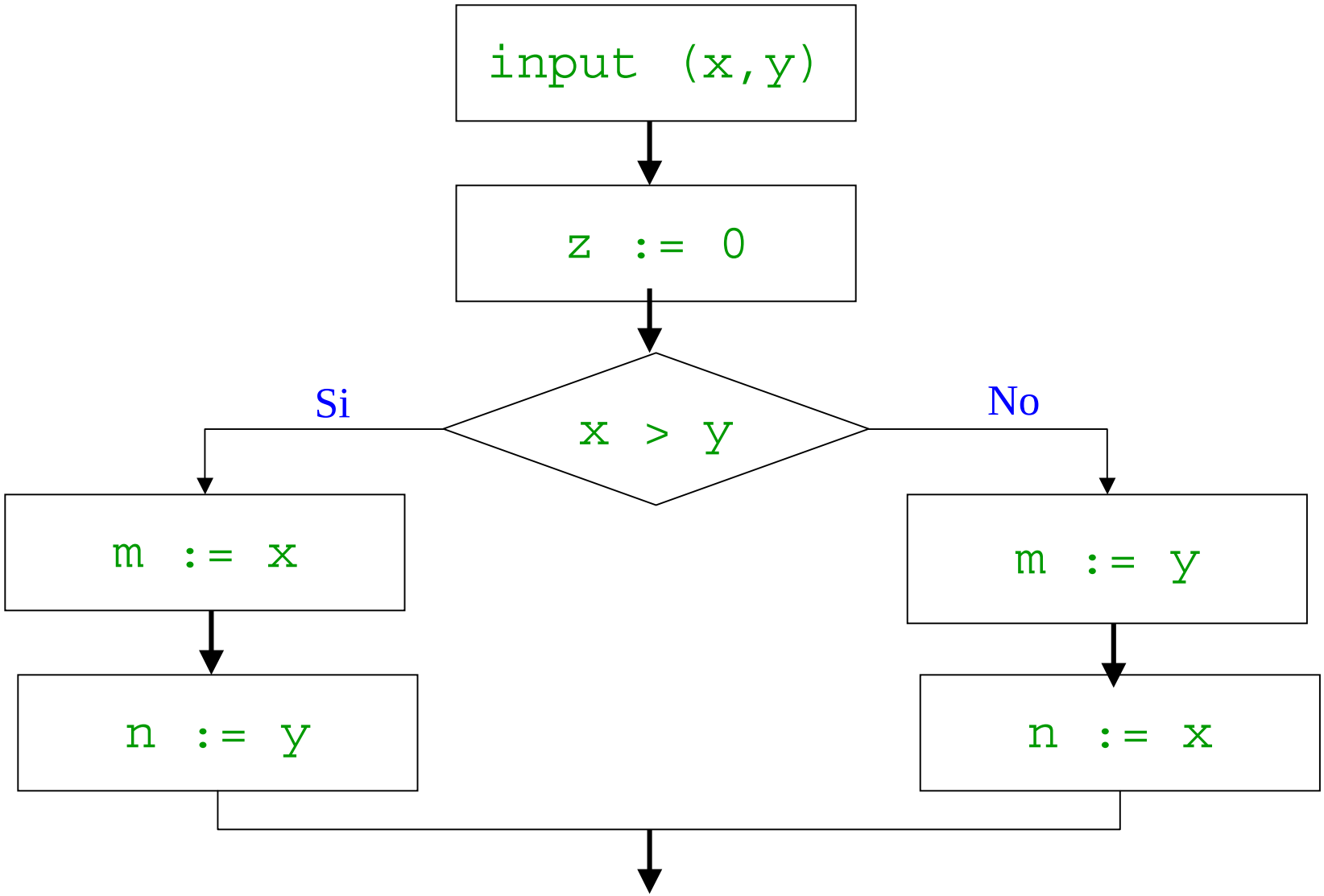
Ragioniamo ora sugli aspetti di efficienza della soluzione fin qui adottata

Il numero di addizioni complessive che vengono eseguite è pari ad y

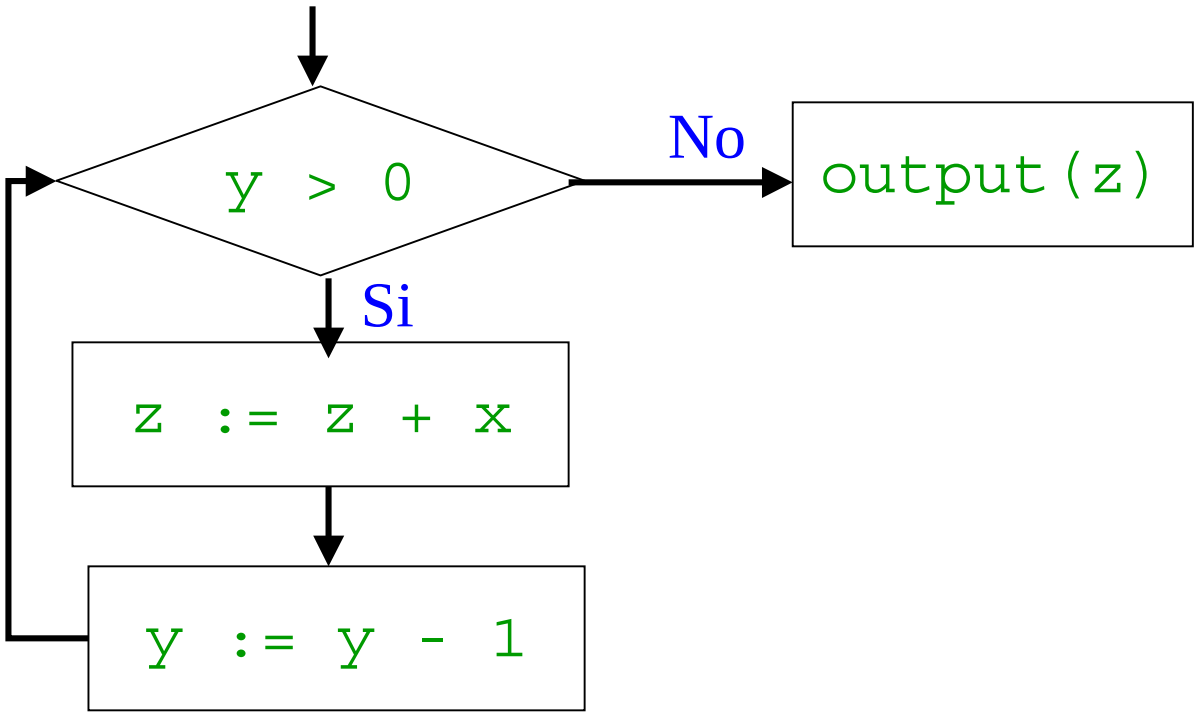
Pensiamo allora al caso di moltiplicare 3 per 17 utilizzando il nostro programma

Forse dobbiamo fare delle modifiche . . .

Diagramma a blocchi



. . . Continua



Efficienza

```
Prodotto
        integer x, y, m, n;
        input (x, y);
        if (x > y) {
              m := x; n := y;
        } else {
              m := y; n := x;
        };
        z := 0;
        while (n > 0) {
              z := z + m;
              n := n - 1;
        }
        output (z);
```

Tipi di istruzioni

- **Input / Output**
- **Assegnazione**
- **Operazioni aritmetiche e logiche**
- **Controllo**
    - **a) Sequenza (Composizione)**
    - **b) Selezione (Alternativa)**
    - **c) Iterazione (Ciclo o Ripetizione)**

Esempio prodotto

```
Prodotto
        integer x, y, m, n;
        input (x, y);
        if ( x > y ) {
                m := x; n := y;
        } else {
                m := y; n := x;
        };
        z := 0;
        while ( n > 0 ) {
                z := z + m;
                n := n - 1;
        }
        output (z);
```

intestazione dichiarazioni input/output assegnazione oper.arit. oper.log. selezione interazione

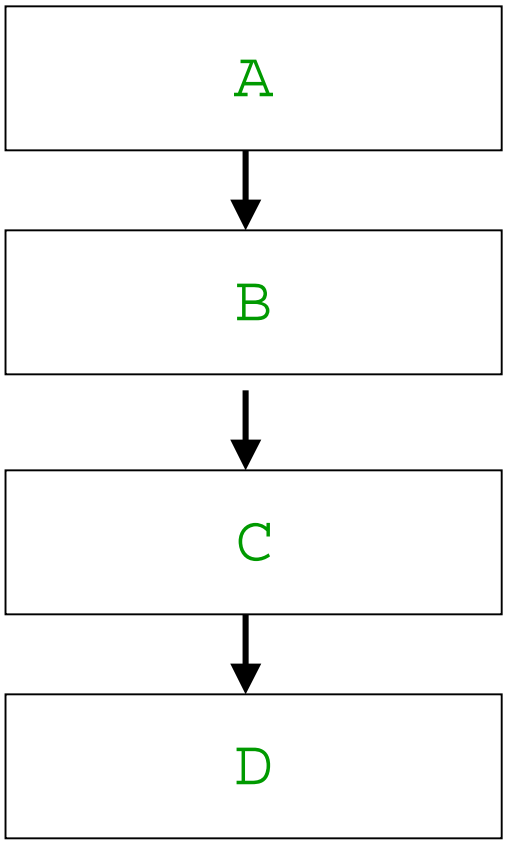
Strutture di controllo

Le strutture di controllo utilizzate nel nostro programma per il calcolo del prodotto di due numeri interi positivi sono

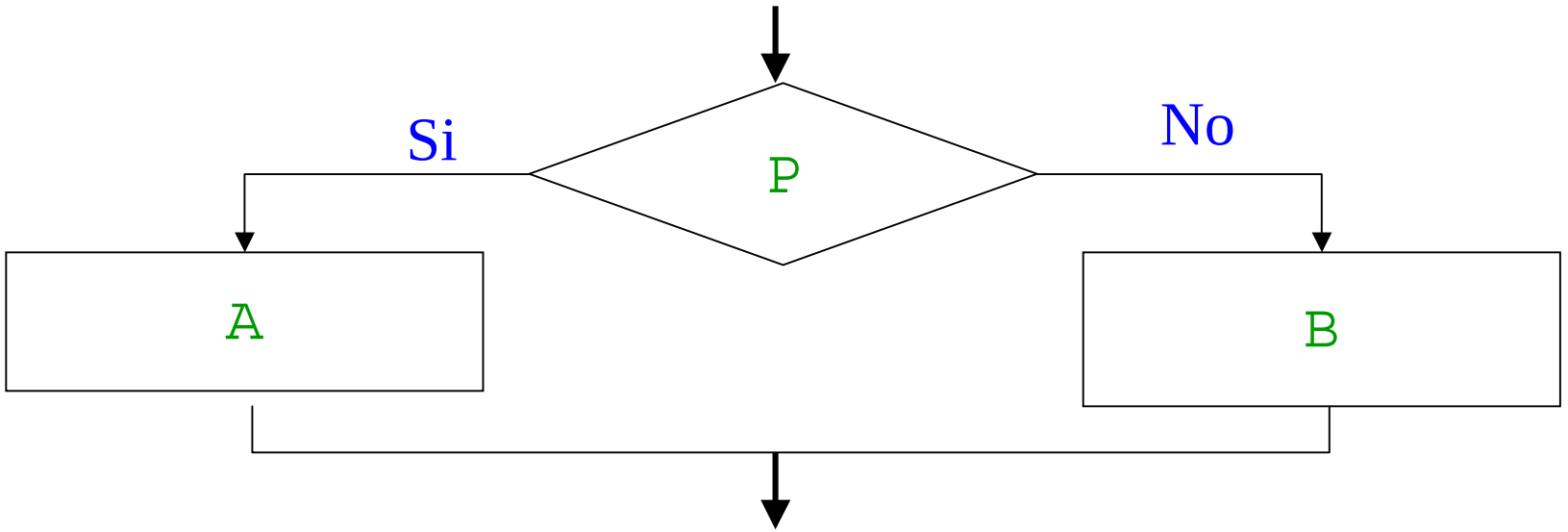
- **Sequenza**
- **Alternativa**
- **Iterazione**

Rivediamole analiticamente attraverso degli schemi diagrammatici

Sequenza

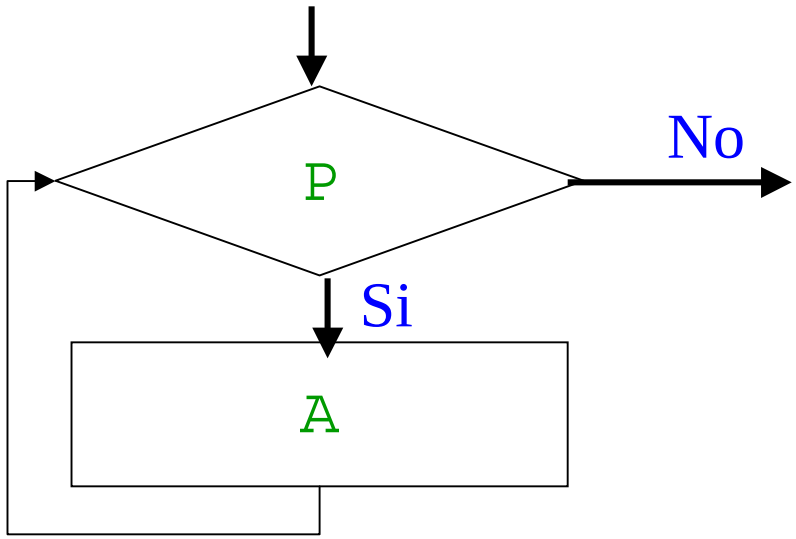


Alternativa



if condizione P istruzione A else istruzione B

Iterazione



while condizione P istruzione A

Programmazione strutturata

Ciascuna delle strutture di controllo precedenti gode della fondamentale proprietà di avere un flusso con una sola entrata ed una sola uscita; possiamo quindi concatenarle tra loro anche annidandole a vari livelli

Qualsiasi programma può essere costruito utilizzando esclusivamente questi tre tipi di strutture di controllo

Ovviamente c'è un teorema su tutto questo

Teorema di Bohm-Iacopini

Riassumendo

Gli esempi introdotti sono stati volutamente semplici ma ci consentono di trarre delle conclusioni generali

- **Per ogni problema (decidibile) posso costruire, cioè sintetizzare, diversi algoritmi tra i quali sceglierò sulla base della loro efficienza**
- **Per ogni algoritmo posso sintetizzare diversi programmi tra i quali sceglierò sulla base della loro efficienza avendone evidentemente verificata la correttezza**

Vedremo meglio cosa significa tutto questo e soprattutto come si fa

E ancora . . .

- **Le soluzioni dei problemi le posso costruire (attraverso algoritmi e programmi) astraendo da ciò che conosco, ad esempio risolvo il problema del prodotto mediante l'iterazione della somma**
- **Similmente posso risolvo il problema dell'elevamento a potenza intera mediante l'iterazione del prodotto**

Attenzione !!!

- **Non è quasi mai vero che la prima soluzione sia la migliore e soprattutto non pensiamo di automatizzare ciò che faremmo a mano**

E ancora . . .

Riflettiamo su questo problema

Dati 100 numeri interi positivi di 4 cifre ciascuno, è più semplice (più rapido) calcolare la loro somma oppure determinare tra loro il valore massimo?

E ancora . . .

Se per verificare come stanno le cose mi rivolgo ad un esecutore umano o ad un elaboratore ottengo la stessa risposta?