Problema: Calcolo del valore assoluto di un numero

Vediamo alcuni esempi di specifica di algoritmi che utilizzano lo pseudo-linguaggio.

Specifica:

Stato iniziale: $\{ \text{numero} \rightsquigarrow A \}$
Stato finale: $\{ \text{risultato} \rightsquigarrow |A| \}$

dove A indica un (generico) valore intero, diverso da zero.

Algoritmo:

```
if (numero > 0)
        risultato =  numero;
    else risultato =  - numero;
```


Problema: scambiare il valore di due variabili

Specifica:

Stato iniziale: $\{ x \rightsquigarrow A, y \rightsquigarrow B \}$
Stato finale: $\{ x \rightsquigarrow B, y \rightsquigarrow A \}$

Algoritmo:

```
temp =  x;
x =  y;
y =  temp;
```

Problema: scambiare il valore di due variabili (cont.)

- Si noti l'utilizzo di un nome simbolico aggiuntivo (temp): è essenziale per poter effettuare correttamente lo scambio tra i valori associati ai due interi (a causa del carattere distruttivo dell'assegnamento e della sequenzialità dell'esecuzione).
- È facile verificare che una sequenza del tipo

```
x = y;
y = x;
```

non consente, in generale, di scambiare i valori associati a x e y.

Esempio:

Stato iniziale		Stato Intermedio
$\{ x \rightsquigarrow 10, y \rightsquigarrow 20\}$	$x = y;$	$\{ x \rightsquigarrow 20, y \rightsquigarrow 20\}$
Stato intermedio		**Stato Finale**
$\{ x \rightsquigarrow 20, y \rightsquigarrow 20\}$	$y = x$	$\{ x \rightsquigarrow 20, y \rightsquigarrow 20\}$

Problema: ordinare due interi positivi

Specifica:

Stato iniziale: $\{ num1 \rightsquigarrow A, num2 \rightsquigarrow B \}$
Stato finale: $\{ num1 \rightsquigarrow max(A,B), num2 \rightsquigarrow min(A,B) \}$

Algoritmo:

```
if (num1 < num2)
    {
        temp = num1;
        num1 = num2;
        num2 = temp;
    }
```

ATTENZIONE: non basta stabilire il massimo e il minimo; occorre anche che, alla fine, il massimo sia associato a num1 e il minimo a num2.

Problema: Elevamento a potenza

Specifica:

Stato iniziale: $\{$ base $\rightsquigarrow$ A, esponente $\rightsquigarrow$ B $\}$ con A$>$0 e B $\geq$0
Stato finale: $\{$ risultato $\rightsquigarrow$ $A^B\}$

- L'algoritmo si basa sulla seguente, ben nota, definizione di elevamento a potenza.

$$\begin{aligned} A^0 &= 1 \\ A^B &= \underbrace{A \times A \times \ldots \times A}_{B \text{ volte}} \qquad (\text{se } B > 0) \end{aligned}$$

Algoritmo:

```
risultato = 1;
while (esponente > 0)
{
    risultato = risultato * base;
    esponente = esponente - 1;
}
```

- Nell'algoritmo, si ipotizza che i valori iniziali di base ed esponente soddisfino i requisiti della specifica.

Input/Output

- Nei problemi visti fino ad ora non ci siamo preoccupati di acquisire i dati iniziali né di produrre in uscita i risultati finali.

  Esempio: Ordinare due valori interi acquisiti in input e stampare i valori ordinati.

  Algoritmo:

```
Input(num1);
Input(num2);,
if num1 < num2
{
      temp =  num1;
       num1 =  num2;
      num2 =  temp;
}
Output(num1);
Output(num2);
```


Moltiplicazione egizia

Primo algoritmo documentato (Papiro di Ahmes, ca 1650 a.C)

Stato iniziale: $\{a \rightsquigarrow A, b \rightsquigarrow B\}$ con $A,B > 0$

Stato finale: $\{p \rightsquigarrow A*B\}$

Algoritmo:

```
input(a); input(b); c = 0
while (a > 0)
{
    if (a è dispari)
        {c = c + b;}
    a = a/2;
    b = b*2; }
output(c);
```

- la divisione è intera

  a pari $\Rightarrow$ a*b = (a/2)*2b

  a dispari $\Rightarrow$ a*b = (a/2)*2b = (a-1+1)/2 * 2b = ((a-1)/2 + 1/2) * 2b = ((a-1)/2)*2b + 1/2 * 2b = ((a-1)/2)*2b + b

Problema: Calcolare quoziente e resto della divisione tra due numeri naturali non nulli.

Specifica:

Stato iniziale: $\{x \rightsquigarrow A,\ y \rightsquigarrow B\}$ con $A,B > 0$

Stato finale: $\{x \rightsquigarrow A,\ y \rightsquigarrow B,\ q \rightsquigarrow Q,\ r \rightsquigarrow R\}$

con $A = Q \cdot B + R$ e $0 \leq R < B$

Supponiamo che l'esecutore non sappia eseguire direttamente la divisione, ma solo la somma e la sottrazione.

Algoritmo:

```
q = 0;
r = x;
while (r ≥ y)
{
    q = q + 1;
    r = r - y;
}
```


Calcolare il MCD con l'algoritmo di Euclide

- Dati due naturali non nulli si calcola il MCD utilizzando le seguenti proprietà:

$$\begin{aligned} MCD(x,x) &= x \\ MCD(x,y) &= MCD(x-y,y) \quad \text{se } x > y \\ MCD(x,y) &= MCD(x,y-x) \quad \text{se } y > x \end{aligned}$$

Specifica:

Stato iniziale: $\{x \rightsquigarrow A,\ y \rightsquigarrow B\}$ con $A,B > 0$

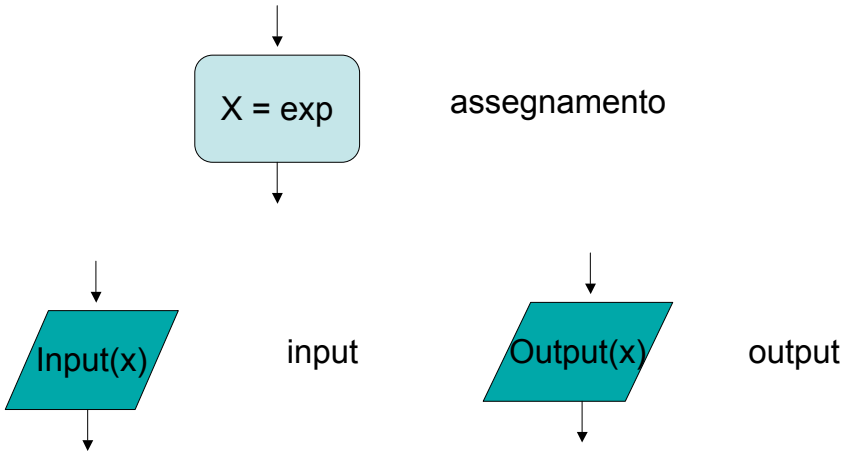
Stato finale: $\{x \rightsquigarrow MCD(A,B)\}$

Calcolare il MCD con l'algoritmo di Euclide (cont.)

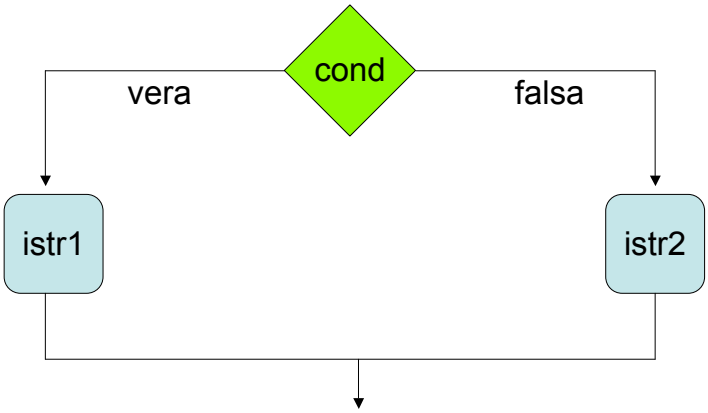
Algoritmo:

```
while (x ≠ y)
{
    if (x > y)
        x = x - y;
    else
        y = y - x;
}
```

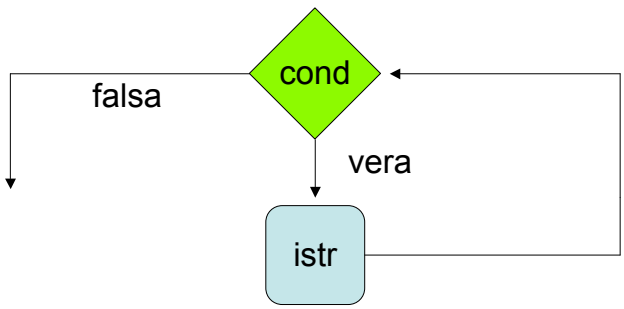

Istr. di controllo:

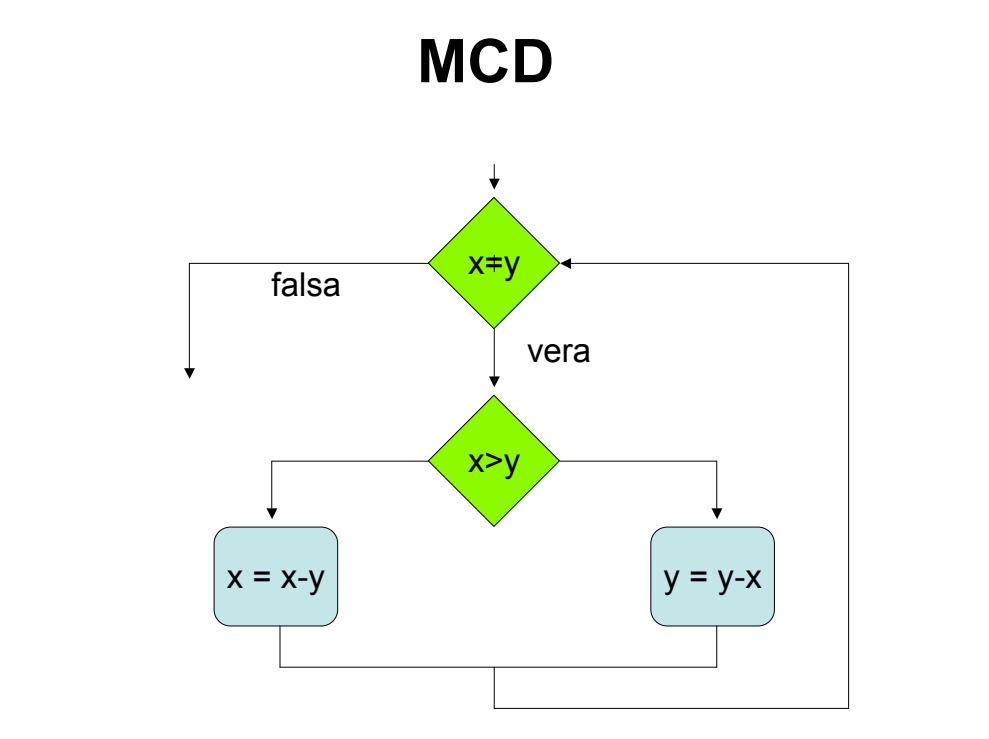


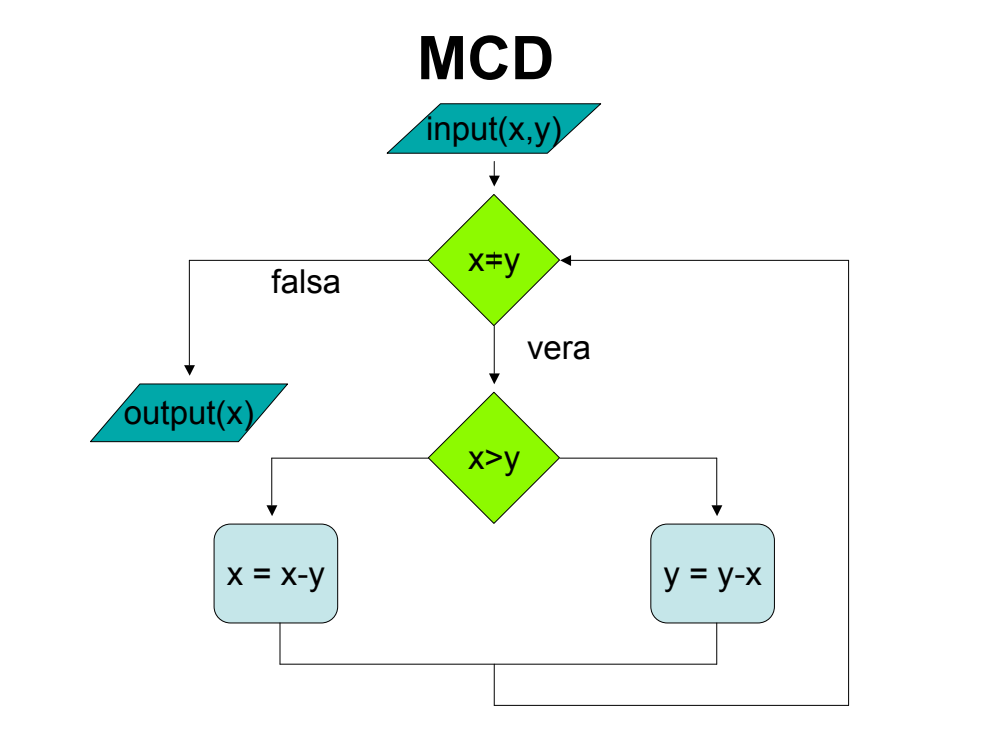
Istr. di controllo: **if** (cond) istr1 **else** istr2

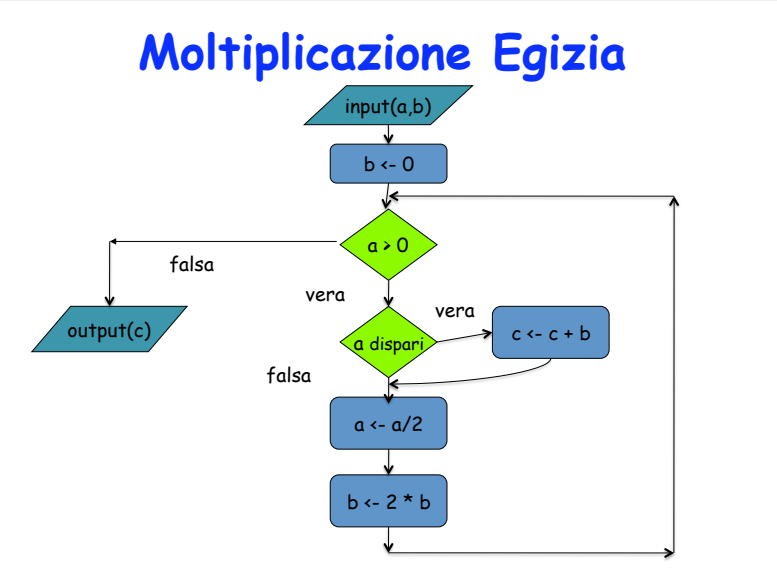


Istr. di controllo: **while** (cond) istr




MCD
x≠y
falsa
vera
x>y
x = x-y
y = y-x


MCD
input(x,y)
x≠y
falsa
vera
output(x)
x>y
x = x-y
y = y-x



Programmazione strutturata

Si parla di programmazione strutturata se si utilizzano solo le seguenti strutture (concatenate in sequenza o annidate una dentro l'altra, ma non intrecciate tra loro), per alterare il flusso del controllo:

- sequenziale
- condizionale
- iterativa

È stato dimostrato che queste tre strutture sono sufficienti per esprimere un qualsiasi algoritmo. Inoltre ci permettono di:

- scrivere programmi facilmente leggibili e modificabili, e di
- evitare l'uso della programmazione a salti (`go to`) che può portare a quello che si definisce dispregiativamente spaghetti code.

Attributi degli algoritmi

Un algoritmo deve essere:

- corretto, ovvero deve fornire un risultato corretto;
- composto con i costrutti più opportuni;
- di facile comprensione, per favorire la manutenzione dei programmi;
- possibilmente elegante;
- efficiente in termini di tempo (quanto lavoro o istruzioni occorrono) e di spazio (quanta memoria si occupa). Solitamente l'efficienza si misura in termini di ordine di grandezza.

Tipi di dato

- per tipo di dato si intende un insieme di valori e di operazioni che si possono ad essi applicare.
- Alcuni tipi di dato (numeri interi, caratteri,etc.), detti tipi primitivi sono forniti direttamente dal linguaggio.
- Si possono tuttavia introdurre nuovi tipi di dato adatti al problema da affrontare.
- Nel caso servano dati aggregati (ad es. una data, un vettore, ecc.), si può infine ricorrere ai tipi di dato strutturati. Si deve poter accedere ai singoli elementi.