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:

```
scanf("%d",&a); scanf("%d",&b); c = 0
while (a > 0)
{
    if (a è dispari)
        {c = c + b;}
    a = a/2;
    b = b*2; }
printf("%d",c);
```

nota che la divisione è intera

a è dispari può essere verificato con la condizione a%2 == 1

Moltiplicazione egizia (cont.)

Algoritmo:

```
scanf("%d",&a); scanf("%d",&b); c = 0
while (a > 0)
{
    if (a è dispari)
        {c = c + b;}
    a = a/2;
    b = b*2; }
printf("%d",c);
```

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

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

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

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

Moltiplicazione egizia (cont.)

Algoritmo:

```
scanf("%d",&a); scanf("%d",&b); c = 0
while (a > 0){
    if (a è dispari) {c = c + b;}
    a = a/2;
    b = b*2;}
printf("%d",c);
```

- $\{a = 16, b = 26, c = 0\}$, $\{a = 8, b = 52, c = 0\}$, $\{a = 4, b = 104, c = 0\}$, $\{a = 2, b = 208, c = 0\}$, $\{a = 1, b = 416, c = 0\}$ $\{a = 0, b = 416, c = 416\}$
- $\{a = 15, b = 26, c = 0\}$, $\{a = 7, b = 52, c = 26\}$, $\{a = 3, b = 104, c = 78\}$, $\{a = 1, b = 208, c = 182\}$ $\{a = 0, b = 416, c = 390\}$

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:

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

Specifica:

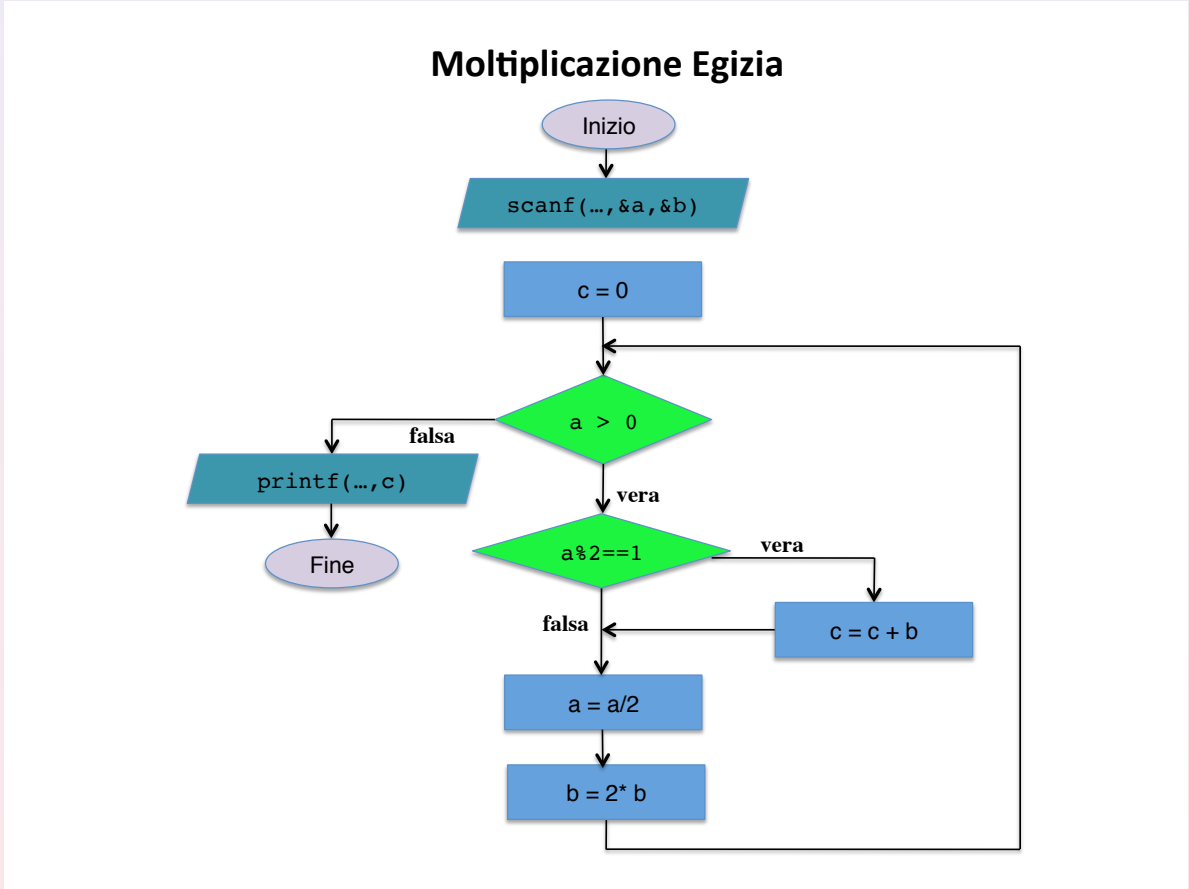
Stato iniziale: $\{x \rightsquigarrow A,\ y \rightsquigarrow B\}$ con $A,B > 0$

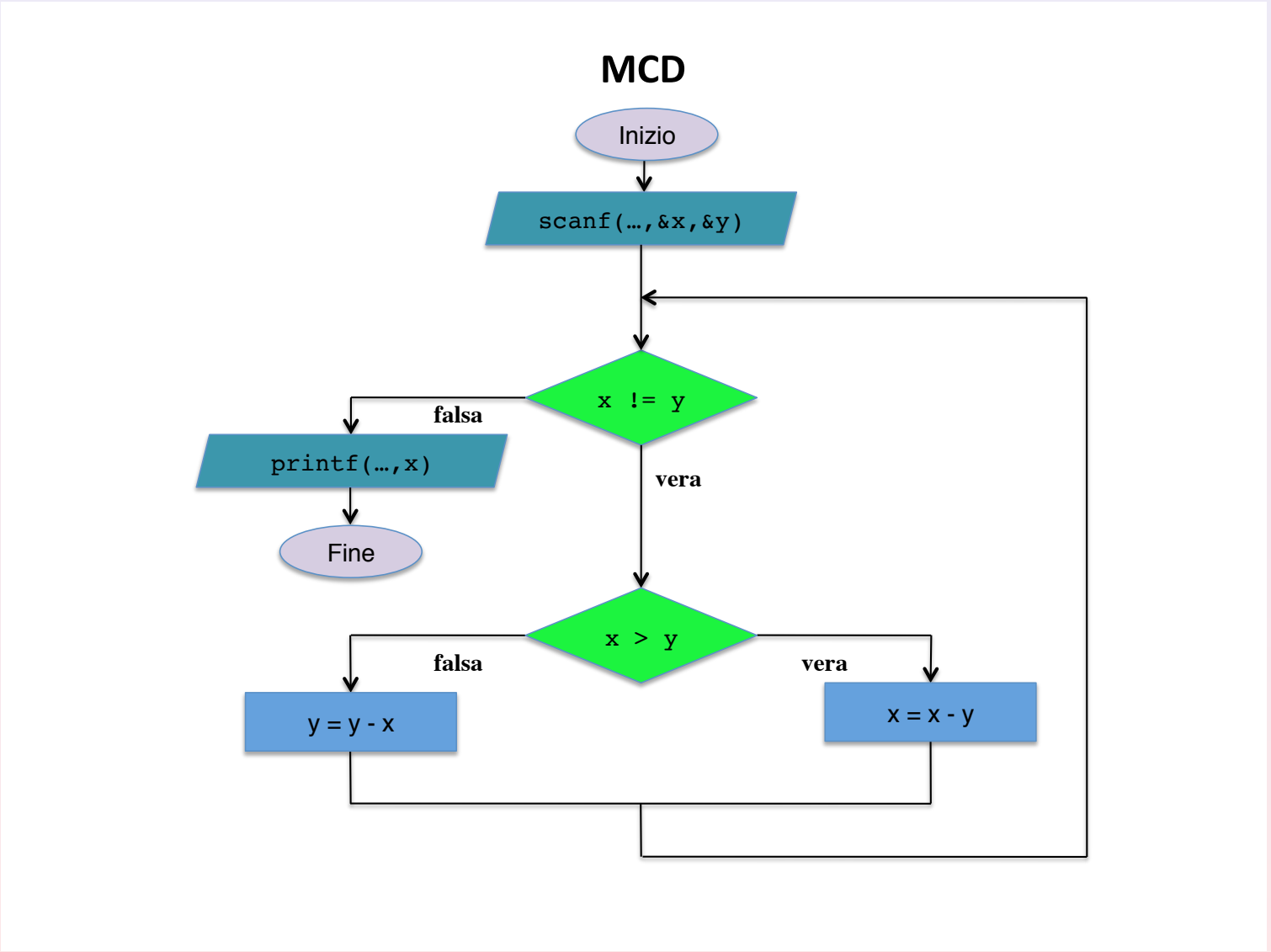
Stato finale: $\{x \rightsquigarrow MCD(A,B)\}$

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

Algoritmo:

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


Moltiplicazione Egizia
Inizio
scanf(…,&a,&b)
c = 0
a > 0
falsa
printf(…,c)
Fine
vera
a%2==1
vera
falsa
c = c + b
a = a/2
b = 2* b


MCD
Inizio
scanf(…,&x,&y)
x != y
falsa
printf(…,x)
Fine
vera
x > y
falsa
y = y - x
vera
x = x - y

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, una programmazione che porta ad una struttura di controllo del flusso complessa e poco comprensibile.

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, ecc.), 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.