14/10/2013

Informatica per la Finanza

Informatica per le Scienze Economiche

Informatica e Programmazione per il Turismo

Programma

- Il corso si divide in quattro parti fondamentali
  - Revisione dei concetti fondamentali di programmazione
  - Il linguaggio VBA
  - Ambiente Matlab e suo linguaggio di programmazione
  - Introduzione alla modellistica

Calendario Corso

- Orario
  - Lu 16:30-19:30 Lab 719
  - Ma 16:30-18:30 Lab 719
- La prossima settimana il calendario
- Prossima lezione 21/10/2013

Riferimenti

- Sito

  digilander.libero.it/lmazzei
- Mail

  luca.mazzei@unimib.it

Modalità esame

– Prova scritta
– Prova orale a discrezione del docente comunicata assieme all'esito delle prove scritte

Algoritmo

- Sequenza ordinata e finita di passi che permettono di risolvere un problema in un tempo finito
  – I passi sono azioni elementari che l'**esecutore** dell'algoritmo è in grado di svolgere (non devono essere ambigui)
  – In un algoritmo vanno anche definiti
    - Gli insiemi dei valori dei possibili ingressi
    - Gli insiemi dei valori delle possibili uscite

Algoritmo

- Gli algoritmi possono essere formalizzati in molti modi fra cui:
  - Diagrammi di flusso (flowchart)
  - Pseudo codifica (solitamente ispirata a qualche linguaggio di programmazione)
- Un semplice programma con cui costruire e provare gli algoritmi è AlgoBuild
  - Si scarica da **`algobuild.com`** (non c'è www)
  - È gratuito e gira in ogni s.o. (è in Java)

Algoritmo: esempi

- Problema: calcolare l'area di un rettangolo
  - Ingressi: i due lati L1, L2 (due numeri positivi)
  - Uscita: A l'area del rettangolo
  - Procedura:
    - Leggere L1,
    - Leggere L2
    - Calcolare A = L1 * L2 (esecutore conosce il prodotto)
    - Mettere in uscita A

Algoritmo: esempi

- Formalizziamo l'algoritmo:

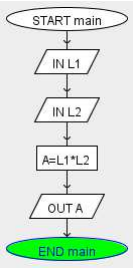


Diagramma di flusso

```
PROG main
    IN L1
    IN L2
    ASSIGN A=L1*L2
    OUT A
END PROG //main
```

Pseudocodifica

Algoritmo: esempi

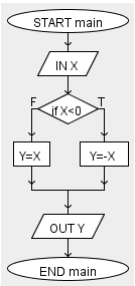
- Una volta scritto l'algoritmo va provato su dei valori in ingresso rappresentativi
  - Occorre
    - prevedere cosa farà l'algoritmo nei diversi passi
    - confrontare il comportamento previsto con quello effettivo
  - Questo non garantisce che l'algoritmo funzioni sempre!

Algoritmo: esempi

- Problema: dato un numero X calcolarne il suo valore assoluto Y
  - Ingresso: numero X
  - Uscita: numero Y
  - Procedura
    - Y = |X| (esecutore conosce operatore valore assoluto)
    - Se X<0 allora Y=-X altrimenti Y = X

Algoritmo: esempi

- Formalizziamo il secondo metodo:




```
PROG main
    IN X
    IF X<0
        ASSIGN Y=-X
    ELSE //if X<0
        ASSIGN Y=X
    END IF //X<0
    OUT Y
END PROG //main
```

Algoritmo: esempi

- Problema: calcolare il prodotto di due numeri naturali
  - Ingresso: due numeri naturali X ed Y
  - Uscita: un numero naturale Z
  - Procedimento
    - Z = X*Y (l'esecutore conosce il concetto di prodotto di due numeri Naturali)
    - E se non conoscesse il prodotto?

Algoritmo: esempi

- Un procedimento si basa sulla definizione di prodotto:
  - Dati due numeri naturali X ed Y il prodotto di X*Y è dato sommando Y volte X
    - La formulazione è un po' oscura

Algoritmo: esempi

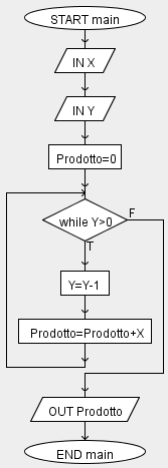
– Supponiamo X=5 Y=4 una procedura potrebbe prevedere questa sequenza temporale:

X	Y	Prodotto
5	4	0
5	3	5
5	2	10
5	1	15
5	0	20

tempo

prodotto = prodotto+X
dopo prima

Algoritmo: esempi




```
PROG main
    IN X
    IN Y
    ASSIGN Prodotto=0
    WHILE Y>0
        ASSIGN Y=Y-1
        ASSIGN Prodotto=Prodotto+X
    END WHILE //Y>0
    OUT Prodotto
END PROG //main
```

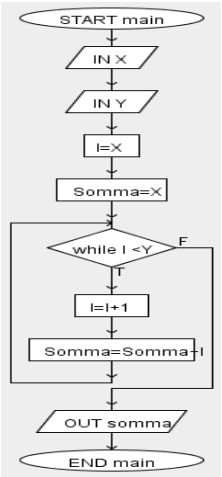
Algoritmo: esempi

- Calcolare la somma di tutti i numeri fra X ed Y (con X<Y)
  - È implicito che X ed Y siano interi
  - Partiamo da un esempio numerico (X=6 Y=10):

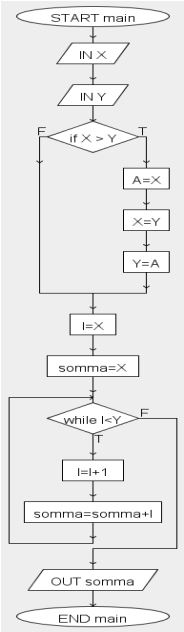
X	Y	I	somma
6	10	6	6
6	10	7	6
6	10	7	13
6	10	8	13
6	10	8	21
6	10	9	21
6	10	9	30
6	10	10	30
6	10	10	40

Notare la variabile I

Algoritmi: esempi




```
PROG main
    IN X
    IN Y
    ASSIGN I=X
    ASSIGN Somma=X
    WHILE I <Y
        ASSIGN I=I+1
        ASSIGN Somma=Somma+I
    END WHILE //I <Y
    OUT somma
END PROG //main
```



Algoritmi: esempi

L'algoritmo precedente modificato in modo che sia sempre X<Y

Algoritmo

- Riepiloghiamo
  - Esistono
    - gli ingressi (variabili)
    - le uscite (variabili)
    - variabili di appoggio (la `I` e la `A` negli esempi precedenti)
  - E' un elenco di istruzioni di diversi:
    - Assegnamento (**=**)
    - Selezione (**`if`**)
    - Iterazione o cicliche (**`while`**)
      - Vi sono più tipi di cicli ma ne basta almeno 1
    - *Chiamata di funzione/procedura (si vedranno in seguito)*
    - *Chiamata Ricorsiva (non ne vedremo)*
      - *La ricorsione sostituisce le istruzioni cicliche*
  - Le istruzioni sono eseguite in sequenza
    - Le istruzioni di selezione e cicliche permettono di eseguire blocchi di istruzioni in base ad una condizione (in pratica alterano la sequenzialità)
  - Un problema può essere risolto in più modi e quindi con algoritmi
  - diversi

Algoritmo: esempi

- Dati due numeri interi positivi A e B calcolare il massimo comun divisore C
  - Metodo delle scuole medie: improponibile, troppo complesso (scomporre il numero in fattori primi non è un algoritmo banale) e lento
  - Metodo basato sulla definizione:
    - Dati due numeri interi positivi A e B il MCD(A,B)=C è il più grande divisore che divide A e B
  - Metodo di Euclide

Algoritmo: esempi

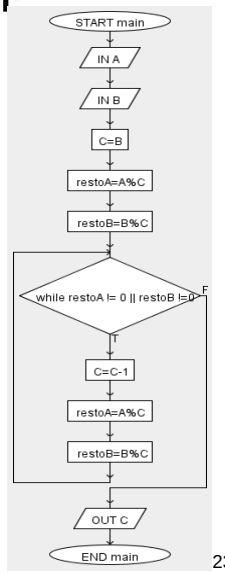
- Metodo basato sulla definizione:

A	B	C	restoA	restoB
35	21			
35	21	21	14	0
35	21	20	15	1
35	21	19	16	2
35	21	18	17	3
35	21	17	1	4
35	21	16	3	5
:				
35	21	8	3	5
35	**21**	**7**	**0**	**0**

	op. resto
%	
&&	AND
||	OR
≠	!=
=	==

Algoritmo: esempi

```
PROG main
    IN A
    IN B
    ASSIGN C=B
    ASSIGN restoA=A%C
    ASSIGN restoB=B%C
    WHILE restoA != 0 || restoB !=0
        ASSIGN C=C-1
        ASSIGN restoA=A%C
        ASSIGN restoB=B%C
    END WHILE //restoA != 0 || restoB !=0
    OUT C
END PROG //main
```



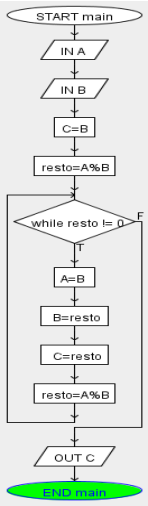
Algoritmo: esempi

- Metodo di Euclide

```
PROG main
    IN A
    IN B
    ASSIGN C=B
    ASSIGN resto=A%B
    WHILE resto != 0
        ASSIGN A=B
        ASSIGN B=resto
        ASSIGN C=resto
        ASSIGN resto=A%B
    END WHILE //resto != 0
    OUT C
END PROG //main
```

A	B	C	resto
35	21	35	14
21	21	35	14
21	14	35	14
21	14	14	14
21	14	14	7
14	14	14	7
14	7	14	7
14	7	7	7
14	7	7	0

=> 7

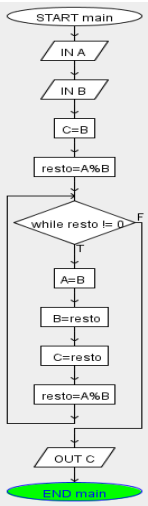


Algoritmo: esempi

- Metodo di Euclide

```
PROG main
    IN A
    IN B
    ASSIGN C=B
    ASSIGN resto=A%B
    WHILE resto != 0
        ASSIGN A=B
        ASSIGN B=resto
        ASSIGN C=resto
        ASSIGN resto=A%B
    END WHILE //resto != 0
    OUT C
END PROG //main
```

A	B	C	resto
3	21	21	3
21	21	21	3
21	3	21	3
21	3	3	3
21	3	3	0
	=>7		

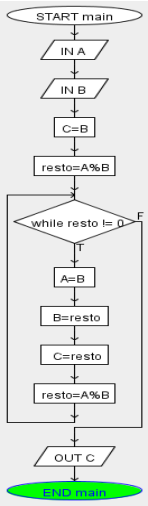


Algoritmo: esempi

- Metodo di Euclide

```
PROG main
    IN A
    IN B
    ASSIGN C=B
    ASSIGN resto=A%B
    WHILE resto != 0
        ASSIGN A=B
        ASSIGN B=resto
        ASSIGN C=resto
        ASSIGN resto=A%B
    END WHILE //resto != 0
    OUT C
END PROG //main
```

A	B	C	resto
21	7	7	0
		=> 7	



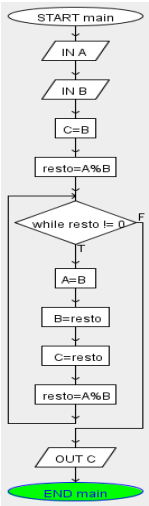
Algoritmo: esempi

- Metodo di Euclide

```
PROG main
    IN A
    IN B
    ASSIGN C=B
    ASSIGN resto=A%B
    WHILE resto != 0
        ASSIGN A=B
        ASSIGN B=resto
        ASSIGN C=resto
        ASSIGN resto=A%B
    END WHILE //resto != 0
    OUT C
END PROG //main
```

A	B	C	resto
21	5	5	1
5	5	5	1
5	1	5	1
5	1	1	1
5	1	1	0

=> 1



Algoritmo: esempi

- La bontà di un algoritmo si giudica:
  - Numero (medio) di passi necessari a risolvere il problema da un input (*complessità in tempo*)
  - Numero (medio) di variabili utilizzate (*complessità in spazio*)
- L'Algoritmo di Euclide per il MCD è migliore del metodo basato sulla definizione

Algoritmo: esempi

- Dato un numero Naturale diverso da 0 N restituire 1 se il numero è primo 0 se non è primo

Algoritmo-Programma

- Programma: traduzione di un algoritmo in un linguaggio di programmazione
  - Il linguaggio di programmazione è un linguaggio che può essere *tradotto* ed eseguito in un elaboratore elettronico
  - È diverso dall'algoritmo perché è vincolato dall'esecutore che:
    - Ha una memoria finita (usa una rappresentazione finita dei numeri ed in generale dei dati)
      - Gli algoritmi pensati per i numeri Interi e reali possono non funzionare

Algoritmo-Programma

- Non esiste un linguaggio di programmazione che permette di tradurre più algoritmi di un altro
- Esistono linguaggi di programmazione:
  - Più facili da apprendere di altri
  - Più versatili di altri
    - Dipende dal contesto di applicazione
- Vedremo:
  - VBA
  - Matlab (come linguaggio di programmazione)