Fondamenti di Informatica

Ingegneria Clinica

Lezione 19/10/2009

Prof. Raffaele Nicolussi

FUB - Fondazione Ugo Bordoni

Via B. Castiglione 59 - 00142 Roma

Docente	**Raffaele Nicolussi**	*rnicolussi@fub.it* *0654803323*
Lezioni Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	**Lunedì, Giovedì, Venerdì**	**12:00 – 13:30**
Esercitazioni Aula 15 Via Tiburtina, 205	**Giovedì**	**14:00 – ???**
Ricevimento:	**Per appuntamento**	**in FUB, per email, per telefono**
Sito web:	**http://w3.uniroma1.it/IngClinFondinf**	

Linguaggio di programmazione

- E' usato per rappresentare le azioni di un algoritmo e la loro concatenazione
  - È composto da tipi di dato (con cui rappresentare i "dati" del programma) e di istruzioni con cui rappresentare le azioni

- *Tipi di linguaggio*
  - Imperativo: fortran, pascal, C
  - Funzionale: Lisp, ML
  - Ad oggetti: Java, C++, SmallTalk

Programmazione imperativa

- E' un paradigma di programmazione secondo cui un programma viene inteso come un insieme di istruzioni (dette anche direttive, comandi)
- Ciascuna istruzione può essere pensata come un "ordine" che viene impartito alla macchina virtuale del linguaggio di programmazione utilizzato.
- Da un punto di vista sintattico, i costrutti di un linguaggio imperativo sono spesso identificati da verbi all'imperativo
- per esempio:
  - 1: input i
  - 2: print i
  - 3: goto 1

Programmazione funzionale

- E' un paradigma di programmazione in cui il flusso di esecuzione del programma assume la forma di una serie di valutazioni di funzioni matematiche.
- Viene usato maggiormente in ambiti accademici piuttosto che industriali.
- Il punto di forza principale di questo paradigma è la mancanza di effetti collaterali (side-effect) delle funzioni, il che comporta una più facile verifica della correttezza e della mancanza di bug del programma e la possibilità di una maggiore ottimizzazione dello stesso.

Programmazione ad oggetti

- La programmazione orientata agli oggetti (OOP, Object Oriented Programming) è un paradigma di programmazione, che prevede di raggruppare in un'unica entità (la classe) sia le strutture dati che le procedure che operano su di esse, creando per l'appunto un "oggetto" software dotato di proprietà (dati) e metodi (procedure) che operano sui dati dell'oggetto stesso.
- La modularizzazione di un programma viene realizzata progettando e realizzando il codice sotto forma di classi che interagiscono tra di loro.

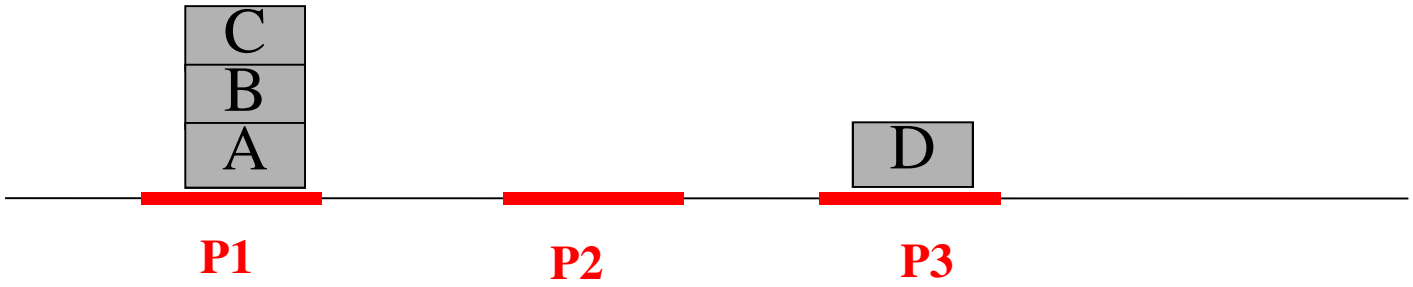
Programma

- *algoritmo scritto in un linguaggio di programmazione al fine di comunicare al calcolatore elettronico le azioni da intraprendere*
- *è la traduzione dell'algoritmo.*
  - Il calcolatore non è in grado di eseguire direttamente il programma che noi scriviamo
  - Il calcolatore capisce solo il linguaggio macchina

Esempio

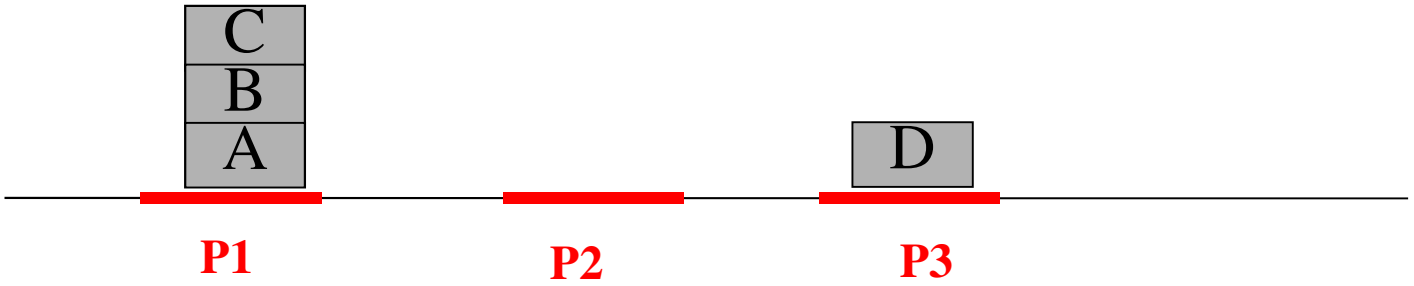
Linguaggio ad alto livello	*Linguaggio macchina*
Sposta il <blocco> in <posizione>	**(put <blocco> <posizione>)**
Sposta <blocco1> su <blocco2>	**(on <blocco1><blocco2>)**

- **Input (dati in ingresso):** **Configurazione iniziale**
  - **A su P1, B su A, C su B, P2 vuoto, D su P3**
- **Problema:** **vogliamo il blocco A sul blocco D**



Esempio

Linguaggio ad alto livello	*Linguaggio macchina*
Sposta il blocco in <posizione>	(put <blocco> <posizione>)
Sposta <blocco1> su <blocco2>	(on <blocco1><blocco2>)



ALGORITMO 1

1. Libera il blocco A
2. Metti il blocco A sul blocco D

PROGRAMMA

(put C P2)
(on B C)
(on A D)

ALGORITMO

1.1 Metti il blocco C in P2
1.2 Metti il blocco B sul blocco C
2. Metti il blocco A sul blocco D

Caratteristiche di un algoritmo

- indipendente dal linguaggio
- deve funzionare per tutte le possibili configurazioni di input per quel problema
- deve potere essere tradotto in un linguaggio di programmazione
- deve essere riusabile

Un algoritmo deve essere:

- **NON AMBIGUO**: le istruzioni devono essere univocamente interpretate dall'esecutore
- **ESEGUIBILE**: l'esecutore deve essere in grado, con le risorse a sua disposizione, di eseguire le istruzioni
- **FINITO**: l'esecuzione dell'algoritmo deve terminare in tempo finito per ogni insieme dei

- **Un algoritmo è una sequenza ben definita e non ambigua di istruzioni che permettono di calcolare i risultati di un problema da parte di un esecutore di istruzioni a partire dalle informazioni a disposizione in tempo finito.**

- **Un programma non è altro che un algoritmo scritto per un calcolatore.**

Problema → **Algoritmo** → **Programma**

Due fasi

1. Analisi dei dati

- *Identificazione dei dati (ingresso e uscita)*
- *Analisi del loro tipo e del loro dominio*
- *Scelta delle modalità di rappresentazione*

2. Specifica della soluzione

- *Scomposizione del problema in sottoproblemi*
- *Identificazione di una soluzione per i sottoproblemi*
- *Identificazione dei passi risolutivi*
- *Concatenazione dei passi*

Algoritmo: somma dei primi 100 numeri

Problema:	**somma dei primi 100 numeri.**
Dati di ingresso:	i numeri **1** e **100**
Uscita:	la *somma* dei **100 numeri**

- **Analisi dei dati**
  - **Ingresso: generazione dei 100 numeri**
    - **i: è una variabile che usiamo sia per generare i numeri che per contare quante volte abbiamo sommato**
    - **i è detto contatore**
  - **Uscita: somma dei primi 100 numeri**
    - **somma: all'inizio vale 0 (zero), alla fine il risultato finale, ma ad ogni iterazione contiene il risultato parziale**

Algoritmo: somma dei primi 100 numeri

Inizializzare *somma* (*somma*=0)
Inizializzare *i* (*i*=1)
Per i che va da 1 a 100
    Aggiungi *i* a *somma* e ponila uguale a *somma*
        (*somma*=*somma*+*i*)
    Incrementa *i* di 1
        (*i*=*i*+1)
Quando *i* =100 stampa *somma*

- La limitazione di questo algoritmo è che funziona solo per i primi cento numeri
- Non è abbastanza generale, ma è
  - non ambiguo
  - termina in tempo finito
  - è eseguibile

Tracciamento

I	SOMMA	Istruzione
?	0	Somma=0
1	0	I=1
1	1	Somma=somma+I
2	1	I=I+1
2	3	Somma=somma+I
3	3	I=I+1
3	6	Somma=somma+I
4	6	I=I+1
4	10	Somma=somma+I
5	10	I=I+1
5	15	Somma=somma+I
6	15	I=I+1
6	21	Somma=somma+I
…		I=I+1
…		

Algoritmo: somma dei numeri da N a M

- **Problema**: **somma dei numeri da N a M**
- **Dati di ingresso**: **i numeri N e M**
- **Uscita**: **la *somma* dei numeri da N a M**

- **Analisi dei dati**
  - **Ingresso:**
    - **N,M**: **lettura dei due numeri**
    - **i**: **per contare da N a M**
  - **Uscita: somma dei numeri**
    - **somma**: **all'inizio vale N, alla fine il risultato richiesto, ma ad ogni iterazione contiene un valore parziale**

Algoritmo: somma dei numeri da N a M

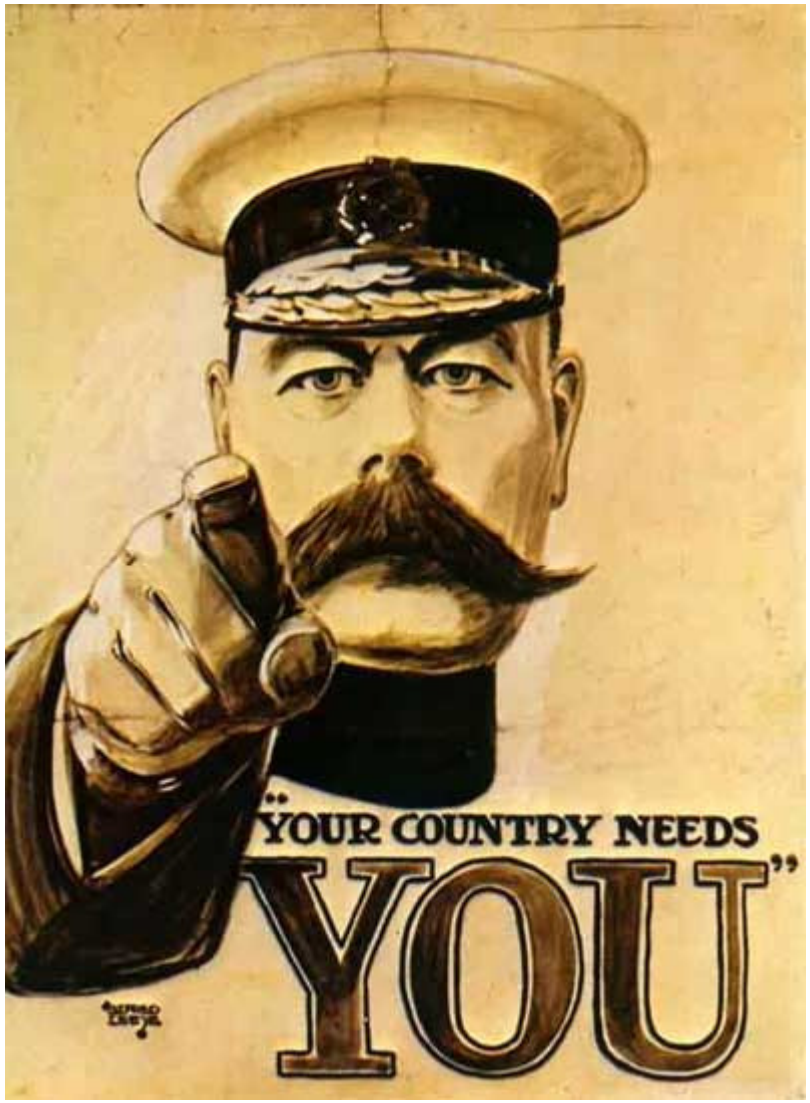
Leggi il numero di partenza *N*
Leggi il numero di arrivo *M*
 Inizializzare *somma* (*somma*=0)
 Inizializzare *i* (*i*=*N*)

 Per *i* che va da *N* a *M*

Aggiungi *i* a *somma* e ponila uguale a *somma*
 (*somma*=*somma* +*i*)
 Incrementa *i* di 1
 (*i*=*i*+1)
 Quando *i*=M stampa *somma*

Abbiamo reso il programma più generale

Ora provaci tu!



Tracciamento con N=3 e M = 6

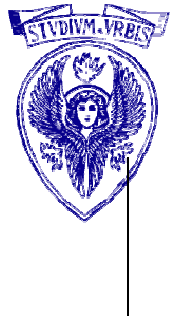
N	M	SOMMA	I	Istruzione
…				

Il processo di risoluzione di un programma

- Algoritmo:
  - Analisi dei dati
  - Specifica della soluzione

- Programma:
  - dichiarazione dei dati
  - specifica della sequenza di istruzioni
    - il modo in cui le istruzioni sono organizzate dipende dal linguaggio (imperativo, funzionale, ad oggetti)

Dati



- I dati sono la materia prima del trattamento dell'informazione
- Si possono elaborare dati di ogni genere
  - numeri, parole, funzioni, grafici, etc.
  - L'importante è trovare la giusta rappresentazione

- Due classi di dati
  - Semplici o primitivi
    - Numeri interi, numeri reali, caratteri
    - Dipendono dal linguaggio
  - Complessi, definiti a partire dai dati primitivi
    - Numeri immaginari, matrici
    - Dipendono dal linguaggio

Dati

- Un dato può essere:
  - costante: rimane immutabile per tutto il programma
  - variabile: cambia valore durante l'esecuzione
- Variabili e costanti hanno un nome, ossia un IDENTIFICATORE
  - N, M sono costanti di tipo intero
  - i, somma sono variabili di tipo intero

Istruzioni: tre classi

- Istruzioni di lettura e scrittura (I/O)
  - Prendi un dato dall'esterno (canali di input, lettura)
  - Manda un dato all'esterno (canali di output, scrittura)
- Istruzioni di assegnazione
  - Assegna un valore ad una variabile
    - I valori assegnati possono essere semplici o espressioni di qualsiasi livello di complessità
    - Non puoi assegnare un valore ad una costante, già ha un valore che non può essere cambiato
- Istruzioni di controllo
  - Determinano il flusso del programma
    - Indicano come le istruzioni sono eseguite
    - C'è ne sono di diversi tipi

Linguaggi: Istruzioni e Dati

- Ogni linguaggio ha sue istruzioni e strutture di dati
  - dipendono dallo scopo per cui è stato progettato il linguaggio

- Alcune istruzioni fanno parte del linguaggio base, altre sono definite a partire da altre istruzioni
  - Il tipo di astrazione dipende dal linguaggio

- La maggior parte dei linguaggi permette astrazioni funzionali
  - L'utente scrive delle funzioni che possono essere riusate più volte all'interno del programma (o anche esportate in altri programmi)
    - le "famose" **black-box**

Pseudo-codice

- Linguaggio artificiale per scrivere algoritmi in modo non ambiguo

Tipi di istruzioni dello pseudo-codice

1. Istruzioni di lettura e scrittura

- Scambiano dati tra il programma e l'ambiente esterno

2. Istruzione di assegnazione

- Modifica i valori delle variabili

3. Istruzioni di controllo

- Istruzione composta
  - Sequenza di istruzioni
- Istruzione condizionale
  - Ramificazione del controllo del programma secondo il valore di una espressione booleana
- Istruzione di iterazione
  - Esegue ripetutamente una istruzione (anche composta) fino a che una data espressione booleana rimane vera