Calcolatori e Algoritmi

Luca Abeni

March 10, 2014

Cos'è un Algoritmo

- Algoritmo: procedura logica (sequenza di azioni) usata per risolvere un problema
  - Può essere espresso anche in modo informale
  - Esempio: ricetta di cucina ("sale quanto basta")
  - Un computer può essere in difficoltà...
- Programma: descrizione formale di un algoritmo
  - Usando un linguaggio di programmazione ben specificato
  - Basso livello vs Alto livello
  - Linguaggio macchina ("01100010...")
- Computer: esegue i programmi (specificati in linguaggio macchina!)

Caratteristiche di un Algoritmo

- Possono essere svariate, a noi ne interessano prevalentemente 2
- Correttezza
  - L'algoritmo risolve il problema per cui è stato pensato
- Efficienza
  - L'algoritmo risolve il problema in poco tempo
  - Esistono definizioni più matematiche e formali...

Esempio

- Problema: Calcolare il fattoriale di n
  - Parametro: n
  - Variabili: i, f
- Algoritmo:
  - Inizializza i a 1 e f a 1
  - Fino a che $i \leq n$
    - $f = f \cdot i$
    - Incrementa i di 1
- Correttezza:
  - Alla fine, $f = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

Esempio - Dettagli

- Un computer non può eseguire direttamente l'algoritmo
- ... L'algoritmo va trasformato in programma
  - Linguaggio di programmazione: permette di esprimere l'algoritmo in modo formale...
  - Esprime un algorimto in maniera rigorosa e precisa
- Linguaggi di alto livello: "vicini" al programmatore, scrivere un programma è più semplice
  - Pascal, C, C++, Java, BASIC, ...
- Linguaggi di basso livello: "vicini" al computer, convertirli in linguaggio macchina (01011110...) è più semplice
  - Assembly

Esempio - Pascal

```
program fattoriale;
 begin
  var f, i, n : integer;

  read(n);
  i := 1; f := 1;
  while i <= n do
   begin
    f := f * i;
    i := i + 1;
   end;
  writeln(f);
 end;
```

- Tutto chiaro, no?

Eseguire il Programma

- Un computer non può capire il Pascal (o linguaggi ad alto livello)
- Il programma va *tradotto* in linguaggio macchina in qualche modo...
  - Sequenza di istruzioni che il computer può capire
  - Dipende dalla macchina...
- Eseguendo le istruzioni macchina, il programma legge un intero in ingresso e stampa il suo fattoriale sullo schermo
- Trasforma un *Input* in un *Output*
- Vedremo più avanti come convertire da linguaggio ad alto livello ad istruzioni macchina...

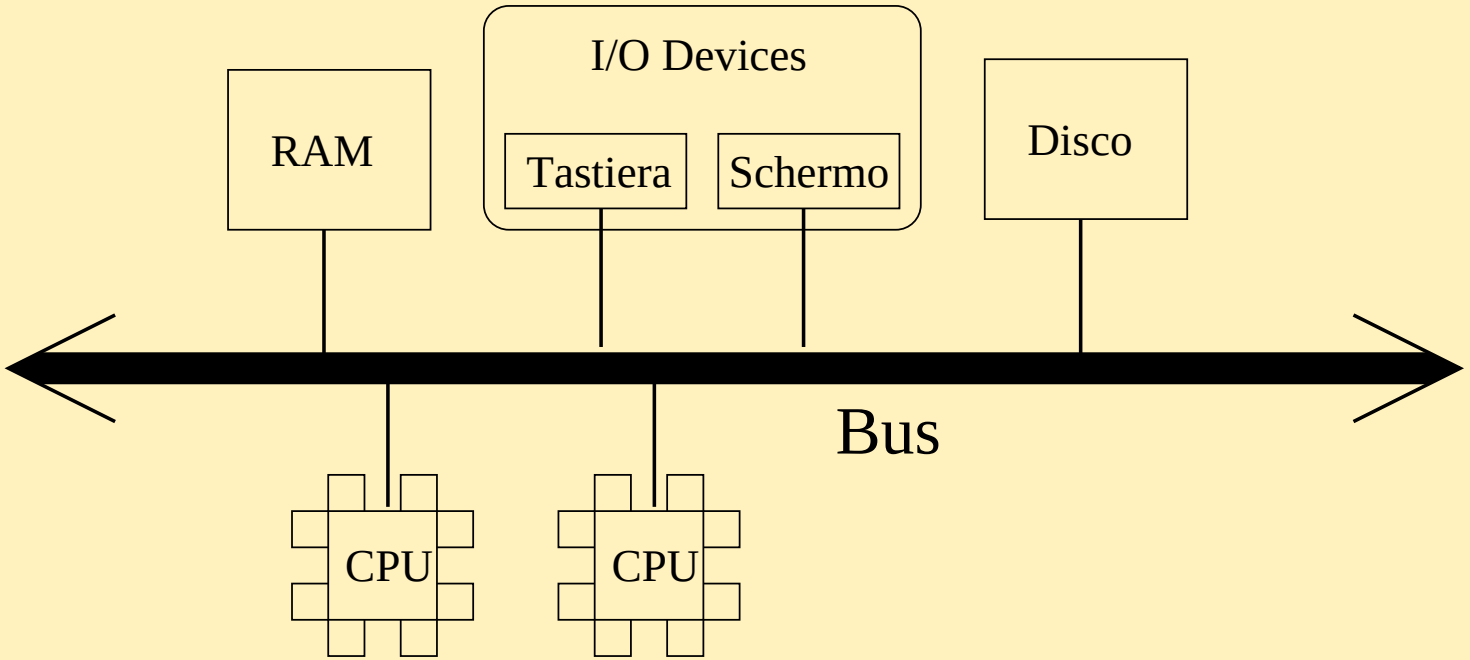
Architettura dei Calcolatori - 1

- Un computer è composto almeno da:
  - Un processore (CPU)
    - Esegue le istruzioni macchina
    - Per fare questo, può muovere dati da/verso la memoria
  - Una memoria centrale (RAM)
    - Memorizza dati e programmi (sequenze di istruzioni macchina)
    - Veloce, ma volatile
  - Una Memoria di massa
    - Più lenta della RAM, ma persistente
  - Alcune periferiche di ingresso/uscita (I/O device)

Architettura dei Calcolatori - 2

- I vari componenti di un computer (una o più CPU, RAM, I/O device, ...) sono connessi fra loro tramite un bus
  - Esempio: bus di sistema
  - Composto da una serie di collegamenti elettrici
- Usato per trasmettere istruzioni macchina e dati fra CPU e RAM
- Usato per Input o Output di dati dai device da memoria di massa

Architettura di Von Neumann



- Memoria che contiene sia dati che istruzioni
- Bus che collega CPU a memoria e dispositivi di ingresso / uscita

Il Processore

- Preleva le istruzioni macchina dalla memoria e le esegue
  - Esecuzione: può aver bisogno di modificare locazioni di memoria (leggere / scrivere dati)
- Parte operativa e parte di controllo
  - Parte di controllo: preleva ed esegue le istruzioni
  - Parte operativa (Arithmetic Logic Unit - ALU): esegue le istruzioni (aritmetiche e logiche)
  - CPU moderne hanno anche una parte floating point (FPU) ed altro...
- Contiene internamente dei registri
  - Visibili od invisibili al programmatore

Registri del Processore

- Invisibili (non acceduti direttamente da istruzioni macchina):
  - Address Register (AR): indirizzo da accedere sul bus
  - Data Register (DR): dato da scrivere / dato letto
- Visibili (menzionati dalle istruzioni macchina):
  - Program Counter (PC): indirizzo della prossima istruzione macchina da eseguire (AKA IP - Instruction Pointer)
  - Status Register (SR): insieme di flag relativi al risultato di operazioni ALU o allo stato della macchina (AKA F - Flag register)
  - Alcuni registri dati ed indirizzi

Esecuzione di un'Istruzione

- Lettura istruzione da eseguire (fetch)
  - Copia PC in AR
  - Trasferisci dato (indirizzato da AR) da RAM a DR
  - Salva DR in un registro invisibile
  - Incrementa PC
- Decodifica istruzione: interpreta l'istruzione
- Esecuzione istruzione: esegui l'istruzione decodificata
  - Se è necessario caricare dati da memoria, setta AR, leggi DR, etc...
  - Se è necessario salvare dati in memoria, setta AR, scrivi DR, etc...
  - Si può modificare PC (jump, etc...)

La Memoria Centrale

- Modello di Von Neumann $\rightarrow$ la stessa memoria può contenere dati o istruzioni
- Acceduta tramite bus
- Insieme di celle (o locazioni) di 8 bit l'una
- Accesso alla memoria:
  - Caricamento indirizzo da accedere nel registro AR
  - In caso di scrittura, caricamento del dato da scrivere nel registro DR
  - Segnalazione (sul bus) dell'operazione da eseguire (lettura / scrittura)
  - in caso di lettura, il dato letto si trova ora in DR

Analisi e Programmazione

- Computer: esegue programmi per risolvere problemi
- Dal problema alla soluzione:
  - Formulazione del problema
  - Sviluppo di un algoritmo che risolve il problema
  - Codifica dell'algoritmo in un programma
  - Conversione del programma in linguaggio macchina
  - Esecuzione del programma (applicato a dati di ingresso)
- Vediamo come eseguire ognuno dei passi precedenti...

Algoritmi

- Ricordi? Algoritmo $\leftarrow$ procedura logica usata per risolvere un problema
  - Spesso espresso come sequenza di azioni
  - Operazioni eseguendo le quali si risolve il problema (meglio: una classe di problemi)
- Matematico arabo del IX secolo: Al Khuwarizmi
- Un algoritmo generalmente si applica a *dati di ingresso* e produce dei *risultati in uscita*
- Azioni generalmente eseguite in ordine sequenziale
  - Il primo passo si applica ai dati di ingresso
  - Altri passi: applicati ai risultati del passo precedente
  - I risultati dell'ultimo passo sono i risultati in uscita

Esempio: il Gioco dell'Undici

- Undici fiammiferi, due giocatori (A e B)
- I giocatori a turno raccolgono da 1 a 3 fiammiferi
- Perde chi è costretto a raccogliere l'ultimo fiammifero
- Esiste un algoritmo che permette al primo giocatore (giocatore A) di vincere sempre
  - A comincia raccogliendo 2 fiammiferi
  - Quando B raccoglie n fiammiferi, A risponde raccogliendone $4 - n$ fiammiferi

Esempio: Ordinamento di un Mazzo di Carte

- Problema: dato un mazzo da 40 carte, ordinare le carte in modo da avere prima cuori, poi quadri, poi fiori e poi picche; le carte di uno stesso seme sono ordinate da 1 a Re
- Come fare:
  - Dividere il mazzo in 4 sottomazzi, ognuno contenente carte dello stesso seme
  - Ordinare ciascun sottomazzo da 1 al Re
  - Mettere i 4 sottomazzi in ordine cuori, quadri, fiori, picche

Esempio: Ricerca di una Chiave

- Trovare in un mazzo la chiave che apre una porta
  1. Selezionare una chiave e marcarla
  2. Verificare se la chiave selezionata apre la porta
  3. Se si, l'algoritmo termina (chiave trovata)
  4. Altrimenti, selezionare una nuova chiave non marcata e tornare al punto 2
  5. Se non si riesce a selezionare una chiave non marcata (tutte le chiavi sono già marcate), la chiave che apre la porta non è nel mazzo

Esempio: Soluzione di un'Equazione di Secondo Grado

- $ax^2 + bx + c = 0$
- Acquisire i coefficienti a, b e c in ingresso
- Calcolare il determinante $\Delta = b^2 - 4ac$
- Se $\Delta < 0$, non esistono radici reali (termina)
- Se $\Delta = 0$, 2 soluzioni reali coincidenti $x_1 = x_2 = -\frac{b}{2a}$. Comunica il risultato e termina
- Altrimenti, $x_1 = \frac{-b+\sqrt{\Delta}}{2a}$ e $x_2 = \frac{-b-\sqrt{\Delta}}{2a}$; comunica i risultati

Esempio: Massimo Comun Divisore di Due Numeri

1. Acquisire i due numeri a e b
2. Se $b > a$ scambia a e b
3. Calcola $r = a\%b$ (resto della divisione intera)
4. Se $r = 0$, $MCD(a, b) = b$; comunica il risultato e finisci
5. Altrimenti:
   - Sostituisci a con b
   - Sostituisci b con r
   - Torna al punto 3

Caratterizziamo un Algoritmo

- Esempi precedenti: espressi in modo diverso...
  - Alcuni più formali, altri meno
  - Alcuni usano notazione più matematica di altri
- ...Ma hanno comunque tutti alcuni aspetti in comune
  - Lavorano (eseguono operazioni) su dati o variabili
  - Interagiscono con l'ambiente esterno (ingresso/uscita)
  - Eseguono operazioni in ordine generalmente sequenziale, salvo eccezioni
    - Eccezioni determinate da condizioni che possono essere vere o false
- Cerchiamo di caratterizzare gli elementi comuni alle descrizioni di algoritmo

Variabili e Costanti

- Le operazioni che compongono l'algoritmo possono agire su *costanti* o *variabili*
  - Costante: dato che non viene aggiornato durante l'esecuzione dell'algoritmo
    - **Esempio:** $\Delta = b^2 - 4ac$
  - Variabile: coppia `(nome,valore)`
    - L'esecuzione dell'algoritmo puo' aggiornare il valore associato al nome della variabile
    - **Esempio:** $\Delta = b^2 - 4ac$
- Una variabile può essere vista come una sorta di contenitore che può contenere un valore

Variabili

- Variabile `(n,v)`: `n` è il nome, `v` è il valore attuale
  - Valore: elemento di un insieme detto insieme di definizione della variabile
  - I valori delle variabili sono specificati solo durante l'esecuzione dell'algoritmo
- **Esempio:** risoluzione equazione di secondo grado $ax^2 + bx + c = 0$
  - a, b e c assumono valori nell'insieme dei numeri reali
  - Nessun valore è assegnato ad essi fino a che l'algoritmo non viene eseguito

Modificare il Valore di una Variabile

- Algoritmo: sequenza di azioni
  - Alcune operazioni possono cambiare il valore di una variabile
  - **Assegnamento**
- **Esempio:** $\Delta = b^2 - 4ac$
  - A sinistra di "=", nome di una variabile; a destra, variabile, costante, o espressione matematica
  - Si valuta l'espressione a destra $(b^2 - 4ac)$
  - Si associa il valore risultante al nome di variabile a sinistra (Δ)

Esempi di Assegnamento

- $a = b + c$
  1. Prima dell'assegnamento, b vale 6 e c vale 4
  2. Esecuzione di $a = b + c$
  3. Dopo l'assegnamento, a vale 10, b vale 6 e c vale 4
- $x = x + 3$
  1. Prima dell'assegnamento, x vale 14
  2. Esecuzione di $x = x + 3$
  3. Dopo l'assegnamento, x vale 17

Variabili Scalari e non

- Le variabili viste finora sono scalari
  - I valori che una variabile può assumere possono essere messi su una scala ordinata
  - Tecnicamente, esiste un ordinamento totale fra i valori che la variabile può assumere
- Esistono anche altre variabili, non scalari
- Variabili composte:
  - Vettori (array, informaticamente parlando) e matrici
  - Strutture
  - ...

Tipi di Operazioni

- Oltre agli assegnamenti che abbiamo visto, un algoritmo contiene altri tipi di azioni (operazioni, istruzioni)
  - Operative: producono risultati
  - di Controllo: controllano il verificarsi di condizioni
  - di Salto: modificano l'ordine di esecuzione
    - Combinabili con azioni di controllo: salto condizionale
  - di Ingresso/Uscita: comunicano con l'ambiente esterno, leggendo dati o scrivendo risultati
- Un algoritmo è descritto indicando come le varie istruzioni di cui sopra modificano i valori delle variabili
- **Esercizio:** Verificare variabili ed azioni (di vario tipo) negli esempi precedenti

Condizioni Vere o False

- Operazioni di controllo si basano su condizioni che possono essere vere o false
  - Proposizioni e Predicati
- Proposizione: "frase" di cui si può dire se è vera o è falsa
  - Ha un valore di verità associato
- Predicato: proposizione il cui valore di verità dipende dal valore di una o più variabili
- **Esempi:**
  - Proposizione: 3 è un numero primo
  - Predicato: il valore della variabile x è un numero primo

Valutazione dei Predicati

- Per stabilire se un predicato è vero o falso, deve essere valutato
  - Sostituzione dei valori delle variabili e determinazione del valore vero o falso
- I valori delle variabili e costanti in un predicato possono essere confrontati usando $<$, $>$, $=$, $\neq$, etc...
  - I risultati di questi confronti possono essere veri o falsi...
- Uhm... Vero, Falso... Ricorda qualcosa?
  - Algebra booleana!!!
  - Le operazioni booleane si applicano a predicati e proposizioni...

Esempi

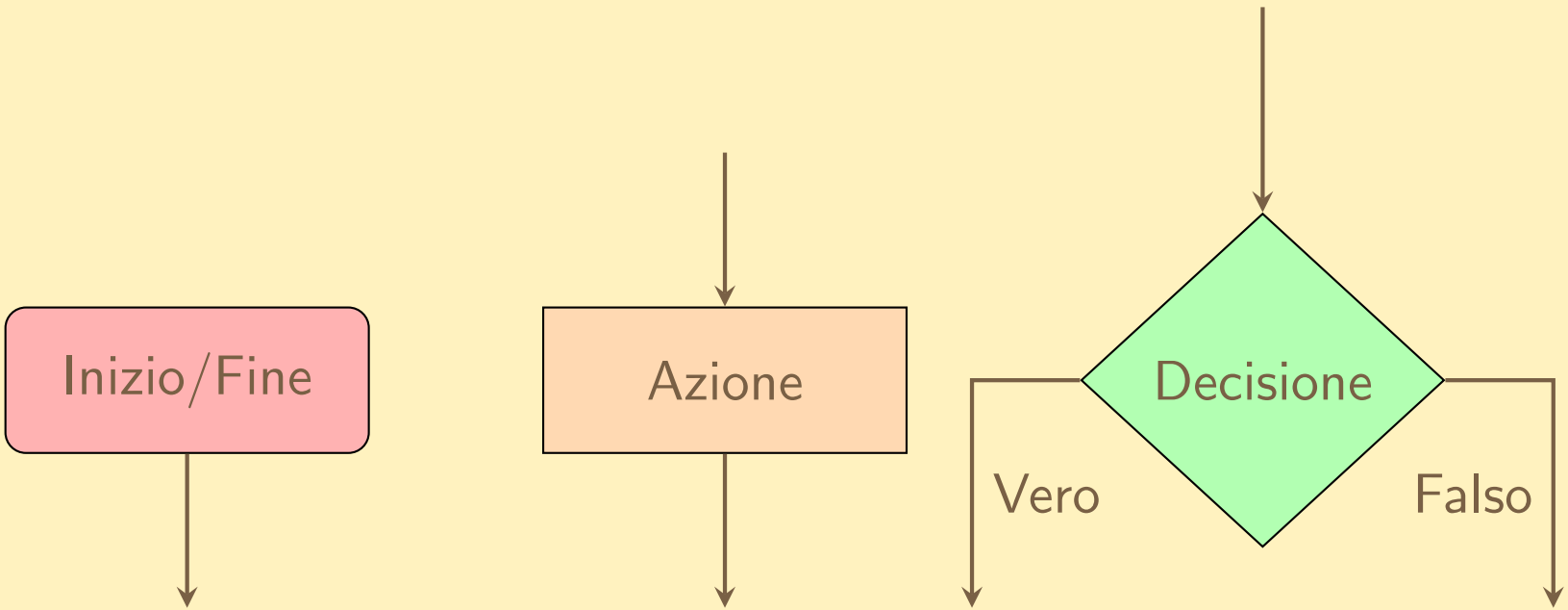
- $\neg(a > b)$
  - Vero solo quando il valore della variabile b è minore od uguale a quello della variabile a
- $a > 30 \wedge a < 50$
  - Vero solo quando il valore della variabile a è compreso fra 30 e 50 (estremi esclusi)
- $a > b \vee a > 100$
  - Vero solo quando il valore della variabile a è maggiore del valore della variabile b o è maggiore di 100

Descrivere un Algoritmo

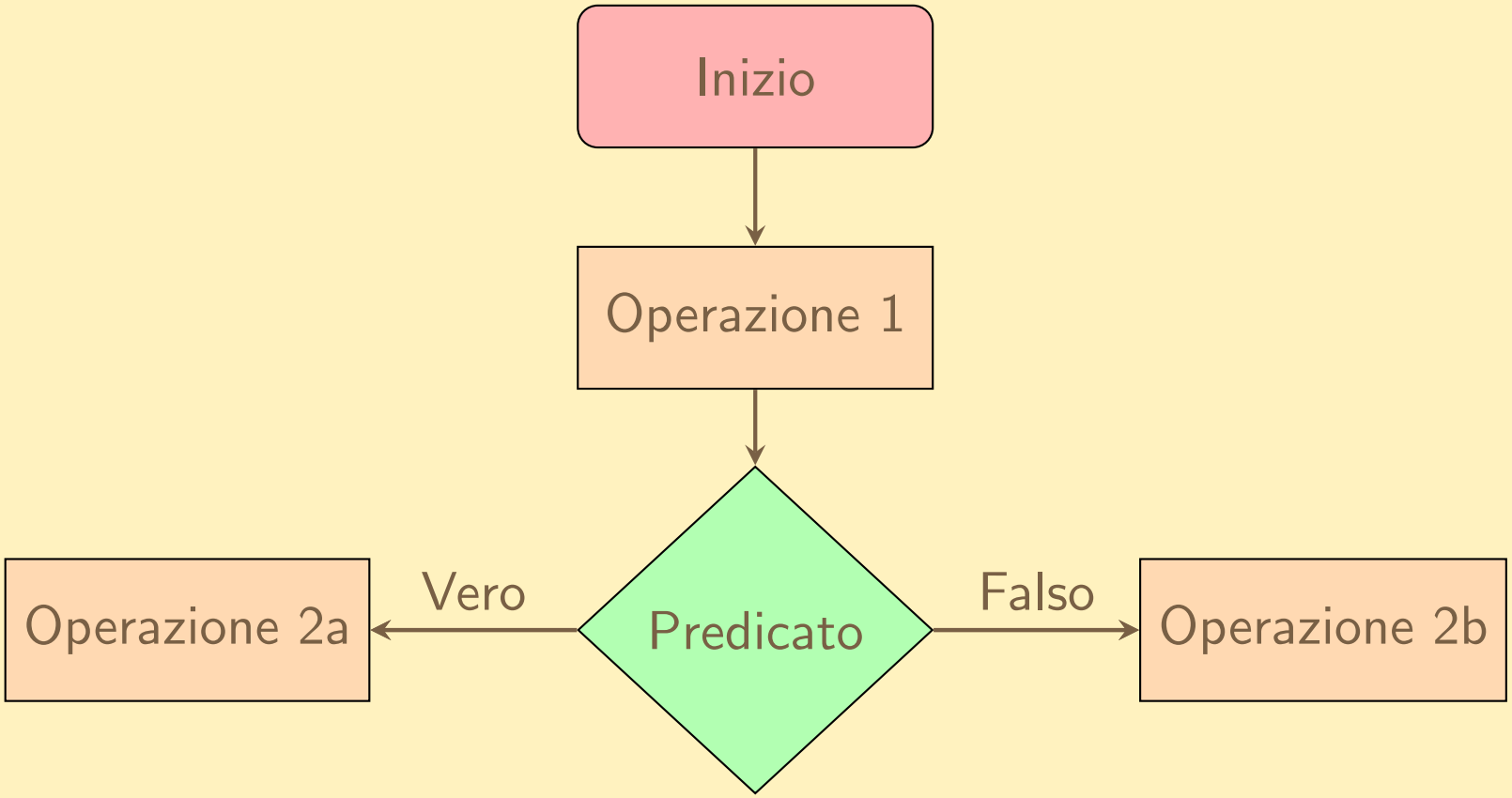
- Algoritmo: composto da operazioni, che agiscono su variabili
- Operazioni di salto possono usare predicati per decidere se effettuare il salto o no
- Come descrivere tutto ciò (in modo formale)?
  - Abbiamo visto varie descrizioni di algoritmi, più o meno formali...
  - Esistono formalismi ben definiti
- Esempio: diagrammi a di flusso (ma esistono anche altri modi)

Diagrammi di Flusso

- Rappresentazione grafica di un algoritmo
- Ogni operazione è rappresentata da un blocco



Diagrammi di Flusso - Esempio



Struttura degli Algoritmi

- Riassumendo:
  - Operazioni eseguite quasi sempre in sequenza
  - Eccezione: operazioni di salto
  - Operazioni di controllo / Predicati: possono decidere quando eseguire determinate operazioni di salto
- Uso scriteriato di operazioni di salto in modo incontrollato $\rightarrow$ difficile seguire il "filo logico" di un algoritmo...
  - Spaghetti programming
- Creazione di una "struttura" nell'algoritmo $\rightarrow$ mette ordine nelle operazioni di salto...

Algoritmi Strutturati

- Usiamo le operazioni di salto in modo "ordinato"!
- Si evidenziano solo 2 casi in cui sono strettamente necessarie:
  - Cicli
  - Selezioni
- Un algoritmo si dice strutturato se è composto solo da:
  - Sequenze di operazioni (esegui `operazione 1`, poi esegui `operazione 2`, ...)
  - Cicli (esegui `operazione` fino a che `predicato` è vero)
  - Selezioni (se `predicato` è vero, allora esegui `operazione 1`, altrimenti esegui `operazione 2`)

Cicli e Selezioni

- Ciclo: ripeti `operazione` fino a che `predicato` è vero
  1. Valuta `predicato`. Se è falso, salta a 4
  2. Esegui `operazione`
  3. Salta ad 1
  4. ...
- Selezione: se `predicato` è vero, esegui `operazione 1`, altrimenti esegui `operazione 2`
  1. Valuta `predicato`. Se è falso, salta a 4
  2. Esegui `operazione 1`
  3. Salta a 5
  4. Esegui `operazione 2`
  5. ...

Operazioni di Salto in Algoritmi Strutturati

- Cicli e selezioni implicano operazioni di salto
- Questi sono gli unici 2 casi in cui un'operazione di salto è ammissibile in un algoritmo strutturato!
- Spesso, i linguaggi di programmazione forniscono costrutti per "nascondere" le istruzioni di salto
  - Un linguaggio di programmazione strutturato non prevede costrutti di salto, ma solo cicli e selezioni
  - Alcuni linguaggi introducono costrutti di salto (`goto`), ma NON USATELI!

Esempio di Algoritmo non Strutturato

- Stampa i numeri da 1 a 10, escludendo il 5

  1. $i = 0$
  2. Se $i \geq 10$, salta a 7
  3. Incrementa i
  4. Se $i = 5$, salta a 2
  5. Stampa il valore di i
  6. Salta a 2
  7. Fine

- L'istruzione 4 non è descrivibile ne' come ciclo ne' come alternativa...
  - "Rompe" la struttura del programma!

Esempio Rivisto

- L'algoritmo precedente è riscrivibile come

1. $i = 0$
2. Se $i \geq 10$, salta a 7
3. Incrementa i
4. Se $i = 5$, salta a 6
5. Stampa i
6. Salta a 2
7. Fine

$=$

1. $i = 0$
2. Mentre $i < 10$
   - Incrementa i
   - Se i non vale 5, stampa i
3. Fine

- Algoritmo strutturato!!!

Algoritmi Strutturati - Più Formalmente

- Se `p` e `q` sono algoritmi stutturati e `b` è un predicato, allora i seguenti algoritmi sono ancora strutturati:
  - `p; q`: esecuzione in sequenza di `p` e `q`
  - se `b` allora esegui `p` altrimenti esegui `q`: selezione
  - fino a che `b` esegui `p`: ciclo
- Potrebbero servire anche un costrutto "non fare niente" ed un costrutto "termina (con errore)"
  - Nessuna necessità di "termina (senza errori)": un algoritmo termina arrivando all'ultima istruzione
  - "non fare niente" $\leftarrow$ "fino a che `Falso` esegui `p`"
  - "termina (con errore)" $\leftarrow$ "fino a che `Vero` non fare niente"

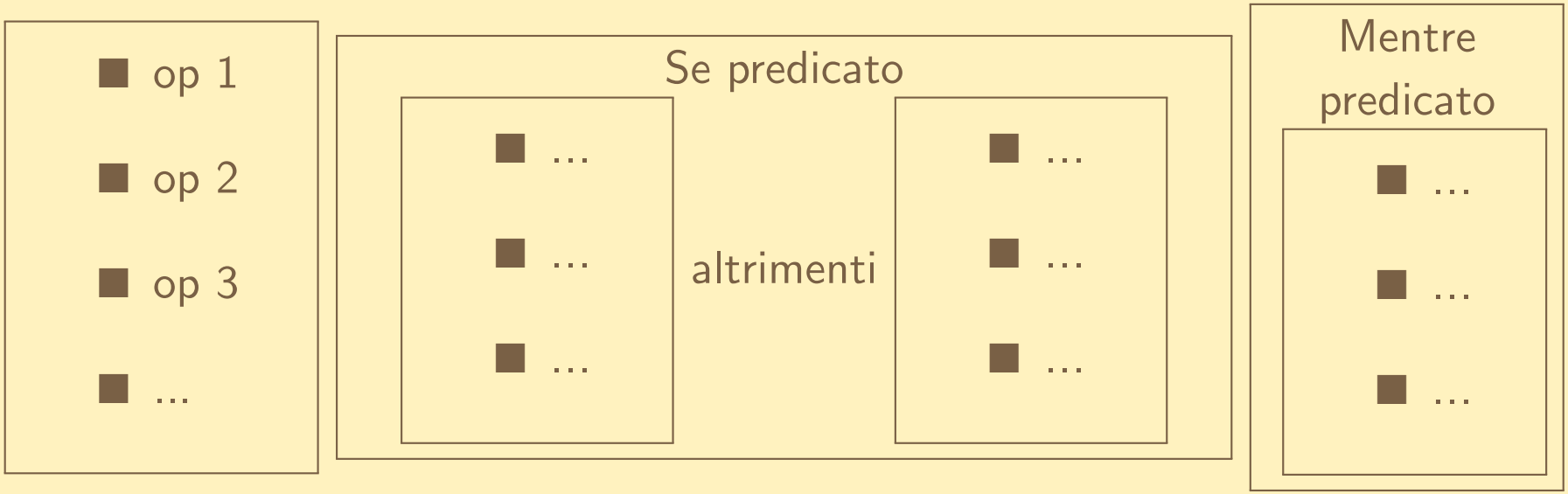
Teorema di Bohm-Jacopini

- Ok, gli algoritmi strutturati sono interessanti...
- ..,Ma siamo sicuri che siano abbastanza "potenti"?
  - Magari rinunciando all'utilizzo "libero" dei salti perdo la possibilità di risolvere alcuni problemi?
- Teorema di Bohm-Jacopini: ogni algoritmo che usa salti (in modo generico) è esprimibile anche come algoritmo strutturato!
  - Se un problema è risolubile attraverso un algoritmo non strutturato, allora è risolubile anche attraverso un algoritmo strutturato!!!
- Questo vuol dire che un linguaggio di programmazione non ha necessariamente bisogno di istruzioni di salto

Proprietà degli Algoritmi Strutturati

- Da cosa deriva il termine "strutturato"?
  - L'utilizzo solo di cicli e cicli e selezioni (senza salti espliciti) da' una "struttura" all'algoritmo
  - Diviene possibile scomporre l'algoritmo in "blocchi di operazioni" che possono essere fra loro concatenati o annidati
- Concatenazione di due blocchi: esecuzione sequenziale prima di un blocco e poi di un altro
- Annidamento di due blocchi: il blocco esterno è una selezione o un ciclo che contiene altri blocchi (2 o 1) al suo interno

Blocchi di Istruzioni



- Sequenza di istruzioni: unico blocco
- Selezione: due blocchi annidati dentro ad uno
- Ciclo: un blocco annidato dentro ad un altro

Ancora sui Cicli

- Ciclo: controllo del predicato in testa o in coda

1. Se `predicato` è falso salta a 6
2. ...
3. ...
4. ...
5. Salta a 1
6. ...

1. ...
2. ...
3. ...
4. Se `predicato` è vero salta a 1
5. ...

- Nel caso 2, il ciclo esegue sempre almeno una volta
- In realtà le due forme sono equivalenti