FONDAMENTI DI PROGRAMMAZIONE
Corso di Laurea in MATEMATICA
a.a. 2021/2022

Chiara Bodei, Roberta Gori, Damiano Di Francesco Maesa

Dipartimento di Informatica
chiara.bodei@unipi.it, roberta.gori@unipi.it, damiano.difrancesco@unipi.it

Obiettivi

Pensare come un informatico o capire come pensa :)

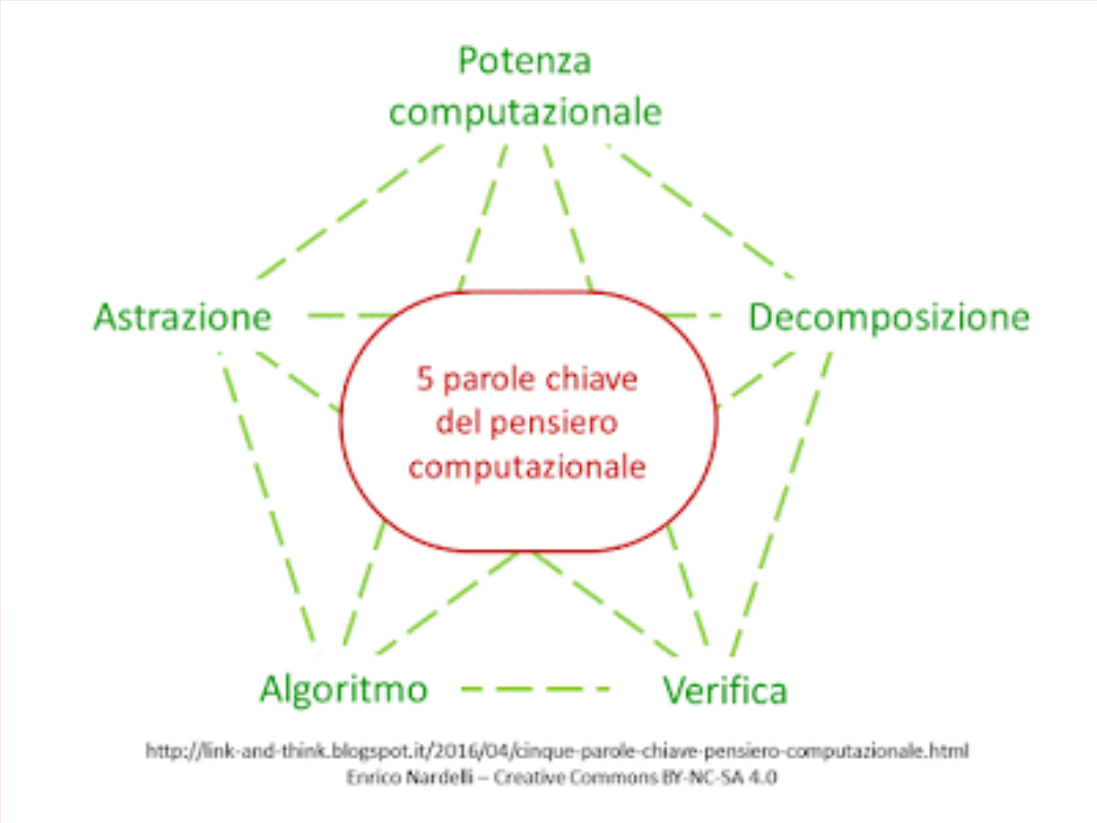
Pensare come un informatico è molto di più che programmare un computer!

> *Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st Century*[1]

- Il "pensiero computazionale" ha a che fare con la risoluzione di problemi (problem solving). Vedi video tratto dall'Apollo 13: `https://www.programmailfuturo.it/progetto/cose-il-pensiero-computazionale`
- Ogni problema si affronta, usando opportunamente l'astrazione e la decomposizione, ricorrendo alle tecniche sviluppate dall'informatica.
- Questo modo di pensare influenza altre discipline: biologia, neuro-scienze, chimica, etc.

[1] J.M. Wing, "Computational Thinking," CACM Viewpoint, March 2006, pp. 33-35.

Pensiero Computazionale in cinque parole-chiave[2]



[2]Vedi l'iniziativa del MIUR e del Cini (il Consorzio Interuniversitario Nazionale per l'Informatica) "Programma il Futuro", pubblicizzata tra l'altro da Marco Belinelli https://programmailfuturo.it/notizie/intervista-marco-belinelli

Pensiero Computazionale

I metodi caratteristici che si acquisiscono con lo studio dell'informatica includono:

- analizzare e organizzare i dati del problema in base a criteri logici;
- rappresentare i dati del problema tramite opportune astrazioni;
- formulare il problema in un formato che ci permette di usare un esecutore (in senso ampio) per risolverlo;
- automatizzare la risoluzione del problema definendo una soluzione algoritmica, consistente in una sequenza accuratamente descritta di passi, ognuno dei quali appartenente a un catalogo ben definito di operazioni di base;
- identificare, analizzare, implementare e verificare le possibili soluzioni con una efficace ed efficiente combinazione di passi e risorse (avendo come obiettivo la ricerca della soluzione migliore secondo tali criteri);
- generalizzare il processo di risoluzione del problema per poterlo trasferire ad un ampio spettro di altri problemi.

Obiettivi (cont)

- Capire come formulare i problemi in modo da poterli risolvere in modo computazionale
- Essere capaci di scrivere programmi di piccola/media dimensione
- Studiare i modelli astratti di calcolo per capire come funzionano i computer e come sono in grado di risolvere i problemi

Informazioni utili

- Docenti: Prof. Chiara Bodei, Jacopo Soldani
- Orario Lezioni: LUN 11-12:45 , MER 14-15:45, GIO 9-10:45
- Orario Laboratorio: (MER 9-10:45): primo laboratorio: MER 6 ottobre
- Ricevimento studenti: Bodei: su appuntamento
- E-mail: chiara.bodei@unipi.it, roberta.gori@unipi.it, damiano.difrancesco@unipi.it

  Pagina web del corso:
  www.di.unipi.it/∼chiara/CORSO_FP_21/FP/index.html

  Testi consigliati per la consultazione:
  - J. Hopcroft-R. Motwani-J. Ullman. Automi, linguaggi e calcolabilità. Addison-Wesley.
  - B.W. Kerninghan, D.M. Ritchie. Linguaggio C. Pearson.
  - Ceri-Mandrioli-Sbattella. Informatica: programmazione. McGraw-Hill

Programma di massima del corso

- Concetti di base della programmazione
- La programmazione nel linguaggio C
- Cenni di teoria degli automi e dei linguaggi

Informatica: cosa è e cosa non è

- **Non** è la scienza e la tecnica dei calcolatori.
- **Non** è lo studio degli utilizzi e delle applicazioni dei calcolatori e del software.
- **Non** è lo studio di come scrivere i programmi per i calcolatori.

È lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione
Association for Computing Machinery.

Algoritmi

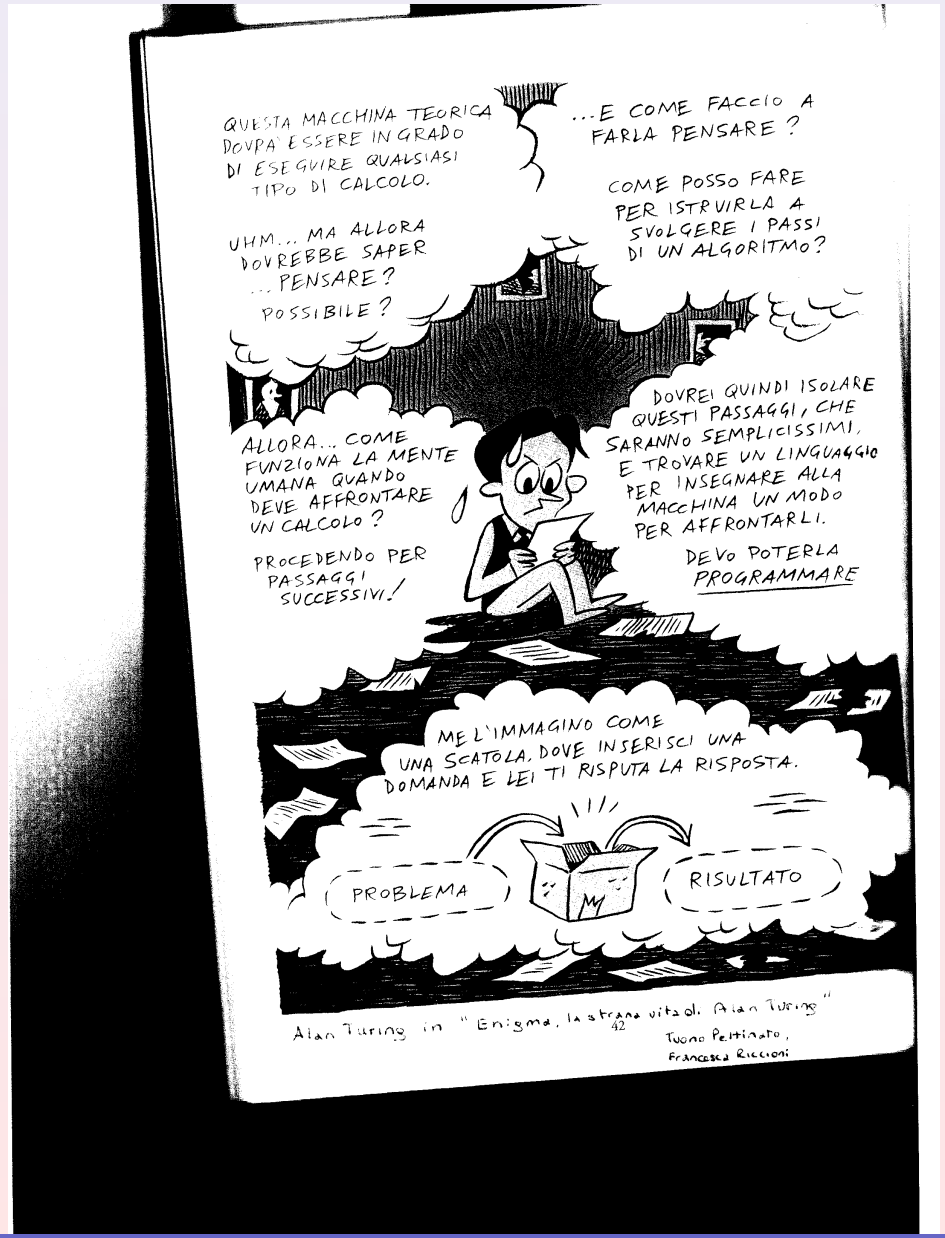
- Utilizziamo algoritmi nella vita quotidiana tutte le volte che, ad es., seguiamo le istruzioni per il montaggio di una apparecchiatura, per impostare il ciclo di lavaggio di una lavastoviglie, per prelevare contante da uno sportello Bancomat, ecc.

> Un algoritmo è una sequenza di passi che, se intrapresa da un esecutore, permette di ottenere i risultati attesi a partire dai dati forniti.

- Una volta in grado di specificare un algoritmo per risolvere un problema, siamo anche in grado di automatizzare il procedimento descritto dall'algoritmo.

Il termine algoritmo deriva dal nome del matematico persiano Muhammad ibn Musa al-Khwarizmi (Corasmia 780 circa - 850 circa). Esercitò la professione nella città di Baghdad, dove insegnava, e introdusse nel mondo arabo i numeri indiani. La sua opera "Il calcolo degli indiani" venne successivamente tradotta in latino da un monaco europeo, con il titolo Liber algarismi - (Il libro di al-Khwarizmi).

Il computer: una macchina per calcolare



Un'idea geniale risolve spesso un grande problema, ma nella risoluzione di tutti i problemi interviene un pizzico di genialità.

Polya G., Come risolvere i problemi di matematica. Logica ed euristica nel metodo matematico. Feltrinelli, Milano, 1967

Quando l'esecutore è il calcolatore

- I calcolatori sono macchine in grado di eseguire velocemente e con precisione sequenze di operazioni elementari.
- Un programma è la descrizione di un algoritmo, espressa in un linguaggio di programmazione che il calcolatore è in grado di comprendere ed eseguire.
- Il calcolatore riceve in ingresso un programma e un insieme di dati iniziali e produce in uscita i risultati dell'esecuzione del programma.
- A differenza di altre macchine automatiche (lavatrici, calcolatrici tascabili, ecc.) i calcolatori sono programmabili: la funzione svolta dipende dal particolare programma che indica alla macchina quali azioni compiere. La macchina non cambia al variare della funzione.

Linguaggi di programmazione

- Esistono moltissimi linguaggi di programmazione che possono essere classificati a seconda del paradigma di programmazione che seguono
- Un paradigma di programmazione indica lo stile e l'insieme di strumenti concettuali da seguire per scrivere programmi, riflettendo un un modo di pensare il processo di computazione.
- Adottando il C seguiremo il paradigma imperativo, per il quale un programma viene come una sequenza di comandi che agiscono sui dati o sull'ordine di esecuzione delle istruzioni [Altri esempi sono: Assembly, FORTRAN, COBOL, Pascal]
- Esistono altri paradigmi di programmazioni, tra i quali: funzionale [ML, Haskell, Scheme], orientato a oggetti [Java, Smalltalk, Eiffel], logico [PROLOG] e concorrente [CCS, CSP].

Cosa intendiamo per programmazione

- Il procedimento che porta alla definizione dei programmi adatti a risolvere problemi è detto programmazione.
- I concetti che stanno alla base della programmazione si possono spiegare e comprendere senza far riferimento al calcolatore.
- La programmazione è tuttavia divenuta una vera e propria disciplina solo con l'avvento dei moderni calcolatori elettronici.

Le fasi della programmazione

Ad un primo livello di astrazione l'attività della programmazione può essere suddivisa in quattro (macro) fasi principali.

1. Definizione del problema (specifica)
2. Individuazione di un procedimento risolutivo (algoritmo)
3. Codifica dell'algoritmo in un linguaggio di programmazione (codifica)
4. Esecuzione e messa a punto (esecuzione)

Specifica

- La prima fase della programmazione consiste nel comprendere e definire (specificare) il problema che si vuole risolvere.
- La specifica del problema può essere fatta in maniera più o meno rigorosa, a seconda del formalismo descrittivo utilizzato.
- La specifica di un problema prevede la descrizione dello stato iniziale del problema (dati iniziali, input) e dello stato finale atteso (i risultati, output).
- La caratterizzazione degli stati iniziale e finale dipende dal particolare problema in esame e dagli oggetti di interesse.

Esempi di specifica informale

1. Dati due numeri, trovare il maggiore.
2. Dato un elenco telefonico e un nome, trovare il numero di telefono corrispondente.
3. Data la struttura di una rete stradale e le informazioni sui flussi dei veicoli, determinare il percorso più veloce da A a B.
4. Scrivere tutti i numeri pari (a parte 2) che non sono la somma di due numeri primi (Congettura di Goldbach [1742]).
5. Decidere per **ogni** programma e per ogni dato in ingresso, se il programma C termina quando viene eseguito su quel dato.

Esempi (contd.)

Caratteristiche comuni ai problemi

informazioni in ingresso $\Longrightarrow$ informazioni in uscita

stato iniziale $\Longrightarrow$ stato finale

Osservazioni sulla formulazione dei problemi:

- la descrizione non fornisce un metodo risolutivo (es. 3)
- la descrizione del problema è talvolta ambigua o imprecisa (es. 2, con Mario Rossi che compare più volte)
- per alcuni problemi non è noto un metodo risolutivo (es. 4)
- esistono problemi per i quali è stato dimostrato che non può esistere un metodo risolutivo (es. 5 - *indecidibili*)

Noi considereremo solo problemi per i quali
è noto che esiste un metodo risolutivo *decidibili*.

Algoritmi

- Una volta specificato il problema, si determina un procedimento risolutivo dello stesso (algoritmo), ovvero un insieme di azioni da intraprendere per ottenere i risultati attesi.
- Il concetto di algoritmo ha origini molto lontane: l'uomo ha utilizzato spesso algoritmi per risolvere problemi di varia natura. Le capacità intellettive necessarie per risolvere un problema sono "codificate" nell'algoritmo.
- Solo in era moderna, ci si è posti il problema di caratterizzare problemi e classi di problemi per i quali è possibile individuare una soluzione algoritmica e solo nel secolo scorso è stato dimostrato che esistono problemi per i quali non è possibile individuare una soluzione algoritmica.

Proprietà di un algoritmo

La descrizione di un procedimento risolutivo può considerarsi un algoritmo se rispetta alcuni requisiti essenziali, tra i quali:

Finitezza: un algoritmo deve essere composto da una sequenza finita di passi elementari.

Non-ambiguità: l'esecutore deve poter interpretare in modo univoco ogni singola azione.

Eseguibilità: il potenziale esecutore deve essere in grado di eseguire ogni singola azione in tempo finito con le risorse a disposizione.

Tipici procedimenti che non rispettano alcuni dei requisiti precedenti:

- le ricette di cucina: *aggiungere sale q.b.* - non rispetta 3)
- le istruzioni per la compilazione della dichiarazione dei redditi (!)

Proprietà di un algoritmo (cont.)

Sarà poi opportuno anche valutare l'efficienza di un algoritmo.

Se dato un elenco telefonico e un nome, per trovare il numero di telefono corrispondente guardo e confronto un nome dopo l'altro, ci metto molto più tempo che procedendo nel modo seguente:

- apro a metà e leggo il nome in cima;
- se trovo il nome che cerco ho finito, altrimenti
- se il nome cercato viene prima in ordine lessicografico, allora cerco nella prima metà;
- altrimenti cerco nella seconda metà;
- andando avanti nello stesso modo fino a trovare il nome o a decidere che non compare nell'elenco.

Quanto tempo ci metterò nei due casi?

Codifica

- Questa fase consiste nell'individuare una rappresentazione degli oggetti di interesse del problema ed una descrizione dell'algoritmo in un opportuno linguaggio noto all'esecutore.
- Nel caso in cui si intenda far uso di un elaboratore per l'esecuzione dell'algoritmo, quest'ultimo deve essere tradotto (codificato) in un opportuno linguaggio di programmazione. Il risultato in questo caso è un programma eseguibile per il calcolatore.
- Quanto più il linguaggio di descrizione dell'algoritmo è vicino al linguaggio di programmazione scelto, tanto più semplice è la fase di traduzione e codifica. Se addirittura il linguaggio di descrizione coincide con il linguaggio di programmazione, la fase di traduzione è superflua.

Codifica (cont.)

I linguaggi di programmazione forniscono strumenti linguistici per rappresentare gli algoritmi sotto forma di programmi che possano essere compresi da un calcolatore.

In particolare dobbiamo rappresentare nel linguaggio di programmazione

- l'algoritmo $\Longrightarrow$ programma
- le informazioni iniziali $\Longrightarrow$ dati in ingresso
- le informazioni utilizzate dall'algoritmo $\Longrightarrow$ dati ausiliari
- le informazioni finali $\Longrightarrow$ dati in uscita

In questo corso impareremo a codificare algoritmi utilizzando il linguaggio di programmazione denominato C.

Esecuzione

- La fase conclusiva consiste nell'esecuzione vera e propria del programma.
- Spesso questa fase porta alla luce errori che possono coinvolgere ciascuna delle fasi precedenti, innescando un procedimento di messa a punto tale da richiedere la revisione di una o più fasi (dalla specifica, alla definizione dell'algoritmo, alla codifica di quest'ultimo).
- Nel caso dei linguaggi di programmazione moderni, vengono forniti strumenti (denominati di solito debugger) che aiutano nella individuazione degli errori e nella messa a punto dei programmi.

Esempi

Negli esempi che seguono, utilizziamo un linguaggio pseudo-naturale per la descrizione degli algoritmi.

Tale linguaggio utilizza, tra l'altro, le comuni rappresentazioni simboliche dei numeri e delle operazioni aritmetiche.

Quanti hanno lo zaino in questa aula?

Specifica

Input: sequenza di postazioni e zaini in aula

Output: il numero di zaini

Algoritmo

Un semplice algoritmo è il seguente:

Passo 1.	Associa zero al *contatore* N
Passo 2.	Per ogni zaino che vedi **ripeti**
Passo 2.1.	Incrementa N di 1
Passo 3.	Ottieni il risultato dell'operazione in N

È corretto?

Quanti hanno lo zaino in questa aula? (cont.)

Specifica

Input: sequenza di postazioni e zaini in aula

Output: il numero di zaini

Algoritmo

Un algoritmo più veloce è il seguente:

Passo 1.	Associa zero al *contatore* N
Passo 2.	Per ogni coppia di zaini che vedi **ripeti**
Passo 2.1.	Incrementa N di 2
Passo 3.	Ottieni il risultato dell'operazione in N

È corretto?

Quanti hanno lo zaino in questa aula? (cont.)

Specifica

Input: sequenza di postazioni e zaini in aula

Output: il numero di zaini

Algoritmo

Un algoritmo più veloce ma corretto è il seguente:

Passo 1.	Associa zero al *contatore* N
Passo 2.	Per ogni coppia di zaini che vedi **ripeti**
Passo 2.1.	Incrementa N di 2
Passo 3.	Se rimane uno zaino
Passo 3.1.	Incrementa N di 1
Passo 4.	Ottieni il risultato dell'operazione in N

È corretto?

Problema 1: Calcolo del prodotto di due interi positivi

Specifica

Input: due valori interi positivi A e B

Output: il valore di A×B

Algoritmo

Se l'esecutore che scegliamo è in grado di effettuare tutte le operazioni di base sui numeri, un semplice algoritmo è il seguente:

Passo 1.	Acquisisci il primo valore, sia esso A
Passo 2.	Acquisisci il secondo valore, sia esso B
Passo 3.	Ottieni il risultato dell'operazione A×B

Problema 1 (cont.)

Codifica

Un esecutore che rispetta le ipotesi precedenti è una persona dotata di una calcolatrice tascabile. La codifica dell'algoritmo in questo caso può essere allora la seguente:

Passo 1.	Digita in sequenza le cifre decimali del primo valore
Passo 2.	Digita il tasto $*$
Passo 3.	Digita in sequenza le cifre decimali del secondo valore
Passo 4.	Digita il tasto $=$

Esecuzione

Problema 1 (cont.)

- Supponiamo ora che l'esecutore scelto sia in grado di effettuare solo le operazioni elementari di somma, sottrazione, confronto tra numeri.
- È necessario individuare un nuovo procedimento risolutivo che tenga conto delle limitate capacità dell'esecutore

Algoritmo 2

Passo 1.	Acquisisci il primo valore, sia esso A
Passo 2.	Acquisisci il secondo valore, sia esso B
Passo 3.	Associa 0 ad un terzo valore, sia esso C
Passo 4.	**Finché** B>0 **ripeti**
Passo 4.1.	Somma a C il valore A
Passo 4.2.	Sottrai a B il valore 1
Passo 5.	Il risultato è il valore C

Problema 1: (cont.)

Un esecutore che rispetta le ipotesi precedenti è un bimbo in grado di effettuare le operazioni richieste (somma, sottrazione e confronto) e di riportare i risultati di semplici calcoli su un quaderno.

Codifica

Passo 1.	Scrivi il primo numero nel riquadro A
Passo 2.	Scrivi il secondo numero nel riquadro B
Passo 3.	Scrivi il valore 0 nel riquadro C
Passo 4.	Ripeti i seguenti passi finché il valore nel riquadro B è maggiore di 0: - calcola la somma tra il valore in A e il valore in C - scrivi il risultato ottenuto in C - calcola la differenza tra il valore in B ed il numero 1 - scrivi il risultato ottenuto in B
Passo 5.	Il risultato è quanto contenuto nel riquadro C.

Cosa si intende con stato

- una particolare configurazione delle informazioni di una macchina, che in qualche modo "memorizza" le condizioni in cui si trova, e che cambia nel tempo passando ad un'altra configurazione, in funzione dei segnali d'ingresso.
- Ad esempio, lo stato può rappresentare la posizione dell'ascensore ad un certo piano di un edificio. In base allo stato si determina il modo in cui l'ascensore si muove: se si intende andare al terzo piano e ci troviamo al primo piano, occorre che l'ascensore salga.
- un sistema è *stateful* se "ricorda' gli eventi precedenti; le informazioni ricordate sono chiamate lo *stato* del sistema.

Il concetto di stato

- La specifica (astratta) di un problema consiste nella descrizione di uno stato iniziale (che descrive i dati del problema) e di uno stato finale (che descrive i risultati attesi).
- Si deve individuare una rappresentazione degli oggetti coinvolti (e dunque dello stato) direttamente manipolabile dall'esecutore.
- Un algoritmo è una sequenza di passi elementari che, se eseguiti, comportano ripetute modifiche dello stato fino al raggiungimento dello stato finale desiderato.
- Le azioni di base che l'esecutore è in grado di effettuare devono dunque prevedere, tra le altre, azioni che hanno come effetto cambiamenti dello stato.
- Possiamo pensare che lo stato sia il contenuto della memoria e che il cambiamento dello stato abbia come effetto il cambiamento di alcune posizioni della memoria.

Un'astrazione dello stato

Dato che il calcolo procede attraverso l'elaborazione dello stato, si devono poter esprimere valori dipendenti dallo stato.

Astrazione del concetto di Stato

Uno stato è un insieme di associazioni tra nomi simbolici e valori.

In uno stato, ad ogni nome simbolico è associato al più un valore.

Rappresentiamo una associazione tra il nome simbolico x ed il valore val con la notazione

$$x \rightsquigarrow val$$

Il valore di x dipende dallo stato corrente, che gli associa un particolare valore val.

Lo stato: esempi

Stati corretti

- $\{\text{nome} \rightsquigarrow \text{Antonio}, \text{cognome} \rightsquigarrow \text{Rossi}, \text{età} \rightsquigarrow 25\}$
- $\{\text{importo} \rightsquigarrow \$1650, \text{tasso} \rightsquigarrow 10\%, \text{interesse} \rightsquigarrow \$165\ \}$
- $\{a \rightsquigarrow 25, b \rightsquigarrow 3, c \rightsquigarrow 50\}$

Stati non corretti

- $\{\text{nome} \rightsquigarrow \text{Antonio}, \text{nome} \rightsquigarrow \text{Paolo}, \text{età} \rightsquigarrow 25\}$
- $\{b \rightsquigarrow 45, a \rightsquigarrow 150, b \rightsquigarrow 10\}$

Prima introduzione al linguaggio C

- Abbiamo visto come un programma non sia altro che un algoritmo codificato in un linguaggio di programmazione.
- Problema: quale linguaggio scegliere per la codifica di un algoritmo?
  - Il linguaggio naturale sarebbe facilmente comprensibile ma non è eseguibile da una macchina.
  - Il linguaggio macchina è eseguibile ma di difficile comprensione.
- Due requisiti fondamentali di un qualsiasi linguaggio per la descrizione di algoritmi:
  - deve essere preciso per non lasciare adito a dubbi interpretativi
  - deve essere sintetico per non rendere difficile la comprensione dei programmi.

- Il linguaggio naturale e il linguaggio macchina si collocano in posizioni opposte, soddisfacendo uno solo dei requisiti.
- I linguaggi di programmazione ad alto livello sono progettati proprio per colmare tale divario.
  $\Longrightarrow$ sono linguaggi adatti a codificare algoritmi pur rimanendo comprensibili.
- La fatica di tradurre un programma nel linguaggio macchina è affidata a particolari programmi, i compilatori, che traducono programmi scritti nel linguaggio di più alto livello in programmi equivalenti nel linguaggio macchina.

Il linguaggio C

- Introduciamo inizialmente l'insieme di costrutti linguistici che costituiscono il nucleo di un qualunque linguaggio di programmazione reale, usando già la sintassi del C.
- Senza entrare in eccessivi dettagli formali, nel presentare le notazioni utilizzate (sintassi) diamo anche una descrizione informale (semantica) di ciò che accade al momento dell'esecuzione in corrispondenza dei vari costrutti.

Il linguaggio contiene costrutti per:

- rappresentare semplici calcoli attraverso le comuni operazioni logico/aritmetiche (espressioni)
- modificare le associazioni nello stato (assegnamento e ingresso)
- controllare l'ordine di esecuzione delle azioni (controllo)
- fornire i risultati (produzione in uscita dello stato finale)

Espressioni

Il ruolo delle espressioni è quello di denotare valori (dip. dallo stato).

Espressioni Numeriche

Il linguaggio consente di rappresentare semplici calcoli algebrici, attraverso le usuali espressioni costruite a partire dai valori numerici e dalle operazioni di

- somma $+$
- sottrazione -
- prodotto *
- divisione intera /
- modulo o resto della divisione intera %.

Il significato di un'espressione è il suo valore ottenuto secondo le usuali regole di calcolo.

Esempio:

- il valore di $3 * 5 + 6$ è 21
- il valore di $3 * (5 + 6)$ è 33

Espressioni (cont.)

Oltre ai valori numerici, le espressioni possono contenere nomi simbolici: il calcolo di un'espressione che contiene un nome simbolico x dipende dallo stato, e precisamente dal valore associato al nome x nello stato.

Esempio:
il valore di $3 * (5 + x)$ è

- 33 in uno stato che contiene l'associazione $x \rightsquigarrow 6$
- 18 in uno stato che contiene l'associazione $x \rightsquigarrow 1$

Gli operatori possiedono regole di precedenze che determinano come avviene la valutazione delle espressioni. Come nell'aritmetica tradizionale, + e - hanno lo stesso grado di priorità, inferiore a quello di *,/ e %.

Espressioni (cont.)

Espressioni Booleane

Si possono rappresentare condizioni ovvero espressioni il cui valore è un valore di verità (true o false).

Le condizioni sono costruite attraverso le usuali operazioni di confronto (==, !=, <, >, <=, >=) e, come nel caso delle espressioni aritmetiche, il loro valore può dipendere dallo stato.
Esempio: il valore di y==x+1 è

- true in uno stato che contiene le associazioni $x \rightsquigarrow 5$ e $y \rightsquigarrow 6$
- false in uno stato che contiene le associazioni $x \rightsquigarrow 5$ e $y \rightsquigarrow 9$

Condizioni più complesse possono essere costruite attraverso operatori logici quali negazione (simbolo !), congiunzione (simbolo &&) e disgiunzione (simbolo ||).

Espressioni (cont.)

- Significato di ! - il valore di verità di ! P è
  - true se il valore di verità di P è false
  - false se il valore di verità di P è true
- Significato di && - il valore di verità di P && Q è
  - true se i valori di verità di P e Q sono entrambi true
  - false altrimenti
  - Se P è false si restituisce false senza valutare Q
- Significato di || - il valore di verità di P || Q è
  - false se i valori di verità di P e Q sono entrambi false
  - true altrimenti
  - Se P è true si restituisce true senza valutare Q
- || and && hanno lo stesso grado di priorità, inferiore a quello di !

Esempio:

- Il valore di `(y >= x) &&(x > 5)` è
  - true in uno stato che contiene le associazioni x $\rightsquigarrow$ 15 e y $\rightsquigarrow$ 30
  - false in uno stato che contiene le associazioni x $\rightsquigarrow$ 3 e y $\rightsquigarrow$ 30
- Il valore di `(y >= x) || (x > 5)` è
  - true in uno stato che contiene le associazioni x $\rightsquigarrow$ 2 e y $\rightsquigarrow$ 30
  - false in uno stato che contiene le associazioni x $\rightsquigarrow$ 3 e y $\rightsquigarrow$ 1