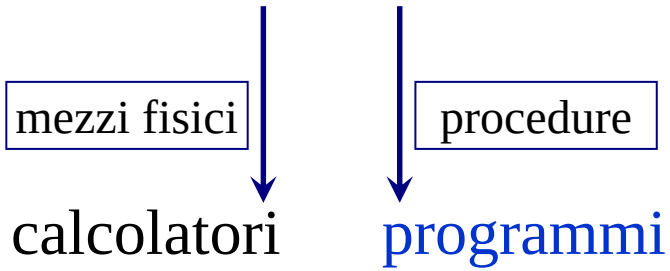
Algoritmi, dati e programmi

Informatica

Informatica:
Scienza che studia l'informazione e la sua elaborazione.

informazione → rappresentazione dell'informazione (dati)

dati → elaborazione dei dati tramite



Traccia

Introduzione ai concetti di:

- problema
- dato
- algoritmo
- linguaggio di programmazione
- programma
- diagramma di flusso

Elaborazione di informazione

Ogni ***problema di elaborazione di informazione*** è caratterizzato da

- un insieme di **dati** di partenza
- un **risultato** cercato

Ogni sua ***soluzione*** è

- una **procedura** che genera il risultato (GIUSTO) sulla base dei dati indicati

Distinguiamo tra:

Competenze distinte

- conoscenza di come si risolve un problema:
  - *analisi* del problema
  - *identificazione* di una soluzione
  - *descrizione* della soluzione
- effettiva capacità di risolvere un problema
  - *interpretazione* della soluzione
  - *attuazione* della soluzione (sviluppo di un programma)

ci permette di sviluppare un programma.

Il programma sarà poi interpretato ed eseguito da un esecutore (essere umano, calcolatore, ...)

Per la descrizione della soluzione si utilizza:

- linguaggio naturale
  - italiano, inglese, …
    - ***sono linguaggi ambigui***
- linguaggio formale
  - formalismo matematico
  - pseudo-codice
  - diagramma di flusso (*flow-chart)*
  - linguaggio di programmazione
  - codice macchina
    - ***sono linguaggi non ambigui***

La descrizione della soluzione può essere fatta a diversi livelli di dettaglio:

Esempio:

- cucinare le lasagne al forno
  - preparare il ragù
  - preparare la besciamella
    - setacciare 50g di farina e stemperarla con 10cc di latte
      - …
    - aggiungere 30cc di latte e 50g di burro a pezzetti
    - ….
  - bollire le lasagne
  - mettere a strati nella teglia e cuocere in forno

Algoritmo

algoritmo [al-go-rìt-mo] *s.m*. Dal Dal nome del matematico arabo Al-Khuwa¯rizmi¯ (sec. IX):

1 (mat.) procedimento sistematico di calcolo: algoritmo algebrico, euclideo.

2 nel Medioevo, il calcolo basato sull'uso delle cifre arabiche.

3 in logica matematica, procedimento meccanico che permette la risoluzione di problemi mediante un numero finito di passi

4 in informatica, serie di operazioni logiche e algebriche, espresse in linguaggio comprensibile all'elaboratore, la cui sequenza costituisce un programma.

Fonte: Dizionario Garzanti on-line (http://www.garzantilinguistica.it)

Algoritmo

La descrizione rigorosa di un metodo che consente di ottenere un risultato attraverso passi elementari si chiama **algoritmo**.

Definizione di algoritmo:

- un insieme *ordinato*
- di passi *eseguibili*
- e *non ambigui*
- che determinano un procedimento atto a risolvere un problema o una classe di problemi
- utilizzando *dati iniziali* e ottenendo dei *risultati*
- in un *tempo finito*.

Esempi di algoritmi

- gli algoritmi per eseguire le 4 operazioni che ci sono stati insegnati alle elementari in un linguaggio adatto ai bambini
- la ricetta per fare le lasagne linguaggio dei libri di cucina

livello di dettaglio diverso a seconda delle competenze dell'esecutore

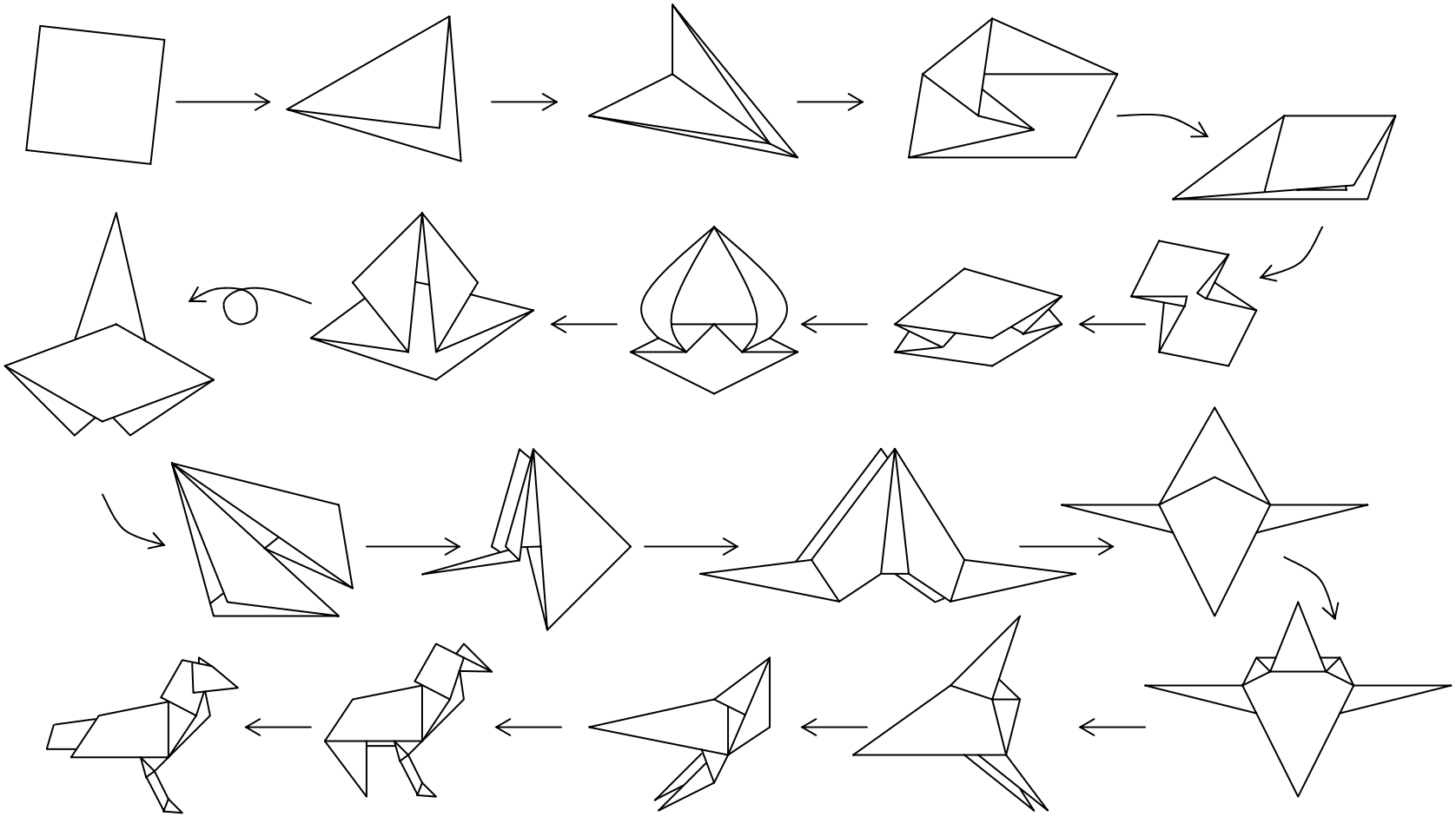
Es.: Algoritmo per accedere a un PC del laboratorio

- Accendere lo schermo se è spento
- Scrivere il proprio <username> nella riga in cui compare la scritta login:
- Scrivere la propria <password> nella riga in cui compare la scritta password
- Se il sistema risponde con la frase: «utente non abilitato», chiamare il tutor

Esempi di algoritmi (2)

12317	7
53	1759
41	
67	
4	

Esempi di algoritmi (3)

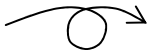
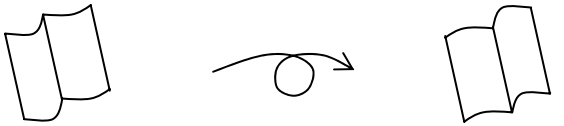
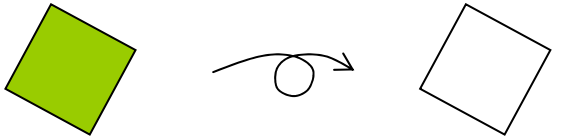
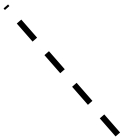
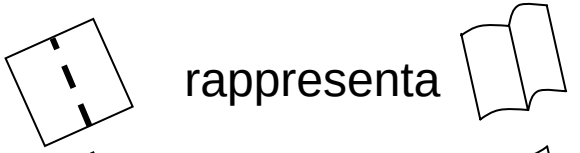
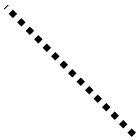
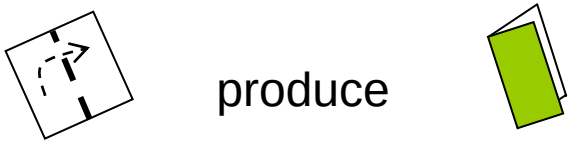
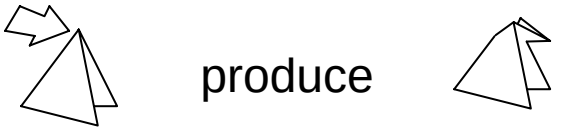


Linguaggio

Un **linguaggio** è costituito da:

- un ***vocabolario;***
- una ***sintassi***, cioè un insieme di **regole** che specificano come comporre i vocaboli per ottenere costrutti ben formati (sintatticamente corretti);
- una ***semantica,*** che associa un **significato** ad ogni costrutto linguistico sintatticamente corretto.

Sintassi e semantica

Sintassi	Semantica	Esempi
	Girare la carta	
Un lato grigio	Distinguere tra i due lati	
	Piegare a monte	 rappresenta
	Piegare a valle	 rappresenta
	Ripiegare	 produce
	Piegare internamente	 produce

Esecutore di algoritmi

Un **esecutore** è un **soggetto** in grado di **attuare** le azioni specificate in un algoritmo

- Nei casi esemplificati, noi

Differenti esecutori possono usare algoritmi diversi per risolvere uno stesso problema

- Nel caso della divisione, un computer NON userebbe l'algoritmo proposto nell'esempio.

Un **esecutore** è quindi caratterizzato da:

- il ***linguaggio*** che è in grado di interpretare
- l'***insieme di azioni*** che è in grado di compiere
- l'***insieme delle regole*** che ad ogni costrutto linguistico sintatticamente corretto associano le relative azioni da compiere.

Programmi

- I **programmi** sono sequenze finite di **istruzioni**, ognuna scritta in un fissato **linguaggio** (**di programmazione**)
- I programmi eseguibili da un computer devono essere scritti usando un linguaggio che il computer è in grado di "comprendere".

Programmi e algoritmi

Un algoritmo può essere quindi specificato sotto forma di programma eseguibile da un calcolatore

Dal problema al programma

Per costruire un programma conviene procedere con metodo

- passando da ***un'analisi del problema*** da risolvere
- ***all'algoritmo di soluzione*** rappresentato in un ***"linguaggio" adatto all'uomo*** ma non troppo lontano dai linguaggi di programmazione
- e ***infine al programma*** scritto nel linguaggio di programmazione prescelto

Il processo di sviluppo di un programma prevede le seguenti fasi:

- ***analisi*** del problema e specificazione dei dati in ingresso e in uscita;
- ***identificazione*** **e** ***formalizzazione*** di una soluzione;
- ***definizione dell'algoritmo*** risolutivo;
- ***programmazione*** in un linguaggio di programmazione "ad alto livello";
- ***traduzione*** in linguaggio macchina;
- ***verifica*** **(*testing*).**

Approccio ***top-down*** alla progettazione:

- si parte da una descrizione ad alto livello della soluzione, in cui si individuano ***sotto-problemi***
- si definiscono le soluzioni dei sotto-problemi in termini di operazioni più elementari
- … e così via, fino ad esprimere tutto in termini di ***problemi elementari***.

Due aspetti da gestire:

- i **dati** da utilizzare
- la successione di **operazioni** da compiere

Linguaggi di programmazione

Un linguaggio di programmazione è costituito da:

- un **vocabolario**
- un insieme di **regole sintattiche** che specificano come comporre istruzioni ben formate
- una **semantica** che associa un “significato” alle istruzioni ben formate, cioè l’azione denotata da ciascuna istruzione.
- Rispetto ad una qualsiasi lingua parlata da esseri umani, un linguaggio di programmazione è molto più semplice, perché la sua sintassi è molto semplice.
- La **teoria dei linguaggi formali** fornisce una classificazione dei linguaggi in base alla loro “semplicità”.

LINGUAGGI DI BASSO LIVELLO

- Linguaggi di programmazione le cui istruzioni corrispondono ad azioni molto elementari. Richiedono uno sforzo di codifica maggiore da parte di un programmatore.

LINGUAGGI DI ALTO LIVELLO

- Linguaggi di programmazione alle cui istruzioni corrisponde un insieme di azioni più articolato. Richiedono uno sforzo di codifica inferiore da parte di un programmatore.

Esempio

Il linguaggio L1 mette a disposizione i comandi:

- Leggi dato A
- Leggi dato B
- Aggiungi_una_unità al dato A
- Esegui per <numero di volte>

Il linguaggio L2 mette a disposizione i comandi:

- Leggi dato A
- Leggi dato B
- Somma <addendo1, addendo2>

Esempio (2)

Vogliamo scrivere un programma per sommare due numeri A e B:

In L1

- Leggi dato A
- Leggi dato B
- Esegui per B volte: Aggiungi _una_unità al dato A

In L2

- Leggi dato A
- Leggi dato B
- Somma (A, B)

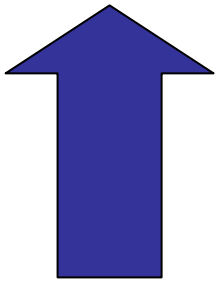
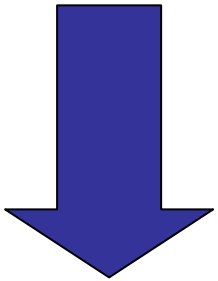
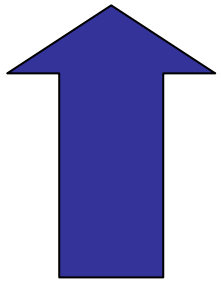
L2 è un linguaggio di livello più alto rispetto a L1, perché offre al programmatore la possibilità di usare istruzioni che sono meno “vicine” al modo in cui lavora il processore e più vicine al modo in cui lavoriamo noi.

LINGUAGGIO MACCHINA

- Il processore è in grado di riconoscere (e quindi di eseguire) solo programmi scritti in un proprio **linguaggio di basso livello** (**linguaggio macchina**).
- Ogni modello di processore (es: Intel, Pentium, Motorola, PowerPC) ha un proprio linguaggio macchina diverso da quello degli altri processori.

LINGUAGGI DI ALTO LIVELLO

- Un programma scritto in un linguaggio diverso dal linguaggio macchina deve essere quindi tradotto nel linguaggio che il processore sa riconoscere.
- I soggetti preposti a questa traduzione sono a loro volta dei programmi (**interpreti** e **compilatori**).

Confronti	Velocità	Portabilità	Complessità
Linguaggio macchina			
Linguaggi di alto livello			

NOTA SULLA VELOCITA'

- Molti linguaggi di alto livello vengono tradotti (*compilati*) in linguaggio macchina da un apposito traduttore (detto compilatore), ottenendo in questo modo delle ottime prestazioni anche in termini di velocità. Esempi: C, PASCAL
- Altri linguaggi sono più lenti poiché vengono tradotti in linguaggio macchina (*interpretati*) al momento dell'esecuzione. Esempi: Matlab

Linguaggi di programmazione: I “contenitori di dati”

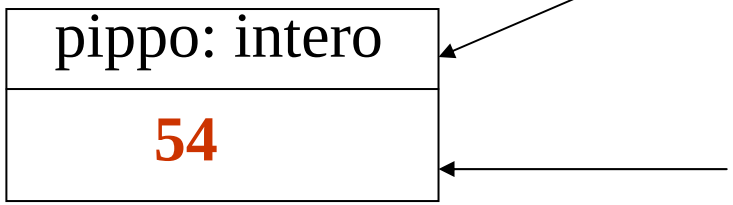
Un contenitore dati

- è un’astrazione della nozione di area di memoria contenente dei dati,
- è detto anche ***variabile*** di programma.

Un contenitore dati ha un ***tipo***, che caratterizza

- un **insieme** di elementi, cioè l’insieme dei valori ammessi per quel contenitore
- le **operazioni** possibili su di essi
- Es.: tipo intero
  - insieme: insieme dei numeri interi rappresentabili
  - operazioni: +, -, *, /, ecc.

Rappresentazione grafica dei contenitori di dati



Nome del contenitore: pippo
tipo : intero

Contenuto = ***dato***
(appartenete al tipo di dati associato al nome, infatti 54 è un numero intero, e su di esso sono ammesse le usuali operazioni aritmetiche)

Tipi di dati

- I linguaggi di programmazione ad alto livello (ma anche Excel e Access) prevedono la tipizzazione dei dati.
- I dati possono essere di **tipo semplice** o di **tipo strutturato**

- I dati di **tipo semplice** hanno un solo valore, ed i tipi comunemente previsti sono:
  - Intero (Es.: -158 0 32)
  - Reale (Es.: 1,414213562373 -56,133)
  - Logico (vero / falso)
  - Testo (Es.: a, pippo, Mario_Rossi, casa)

- I dati di **tipo strutturato** contengono più valori, ad esempio:
  - **vettori** e **matrici** → strutture uniformi, cioè più valori dello stesso tipo (Es.: vettore di interi [2,5,7,3,12,43])
  - **Record** → strutture non uniformi, cioè più valori non necessariamente dello stesso tipo

Es.: studente

cognome : caratteri	Rossi
nome : caratteri	Mario
matricola : intero	656565
….	….

Istruzioni

- Tre categorie:
  - istruzioni di ingresso e uscita (lettura e scrittura)
  - istruzioni aritmetico-logiche (assegnamento)
  - istruzioni di controllo (selezione, iterazione)

Istruzioni di ingresso/uscita

Le istruzioni di **ingresso / uscita** permettono di acquisire dati e di presentare risultati

Esempi:

- **read a** → acquisisci un dato da tastiera e mettilo nel contenitore ‘a’
- **print 'La media dei valori dati in ingresso è ', media** → stampa il contenuto di ‘media’ preceduto da un commento

Istruzioni di assegnamento

- Le istruzione di **assegnamento** modificano lo stato di memoria, cioè i valori dei contenitori dati (variabili). Sono della forma:
  - CONTENITORE = ESPRESSIONE

  → ***metti*** ESPRESSIONE ***in*** CONTENITORE

  Es.: `a = b + 3`

  - ESPRESSIONE può essere: una costante, una variabile, un'espressione vera e propria, una funzione
  - CONTENITORE è il nome di una variabile
  - l'esecutore valuta l'ESPRESSIONE e mette il valore così calcolato in CONTENITORE, sostituendone il valore precedente.

Espressioni

- Le **espressioni aritmetiche** esprimono calcoli numerici
  - somma, sottrazione, prodotto divisione
  - elevamento a potenza, radice, logaritmo, esponenziale, ecc.
    - Es.: b^2 – 4 * a * c

- Le **espressioni sui caratteri** modificano parole e testi.
  - concatenazione
    - Es.: auto&mobile (risultato: automobile)
  - …

- Le **espressioni logiche** (o ***booleane***) esprimono calcoli logici e possono quindi assumere solo i valori *vero* o *falso.*
  - Operatori logici relazionali: confronto fra due valori, ad es.:
    - x < y
    - (x + 5) == y
    - ecc.
  - Operatori logici AND, OR, NOT, ecc. per comporre, ad es.:
    - (x < y) AND (y < z)
- Un valore booleano (vero, falso) è rappresentabile con un bit. Convenzionalmente si assegna 1 a vero, 0 a falso.

Le espressioni di tipo *booleano* assumono particolare importanza perché, usate nelle istruzioni di controllo, ci permettono di prendere delle **decisioni**.

Istruzioni di controllo

Le istruzioni di **controllo** permettono di modificare il flusso di esecuzione delle istruzioni all'interno di un programma, altrimenti puramente sequenziale.

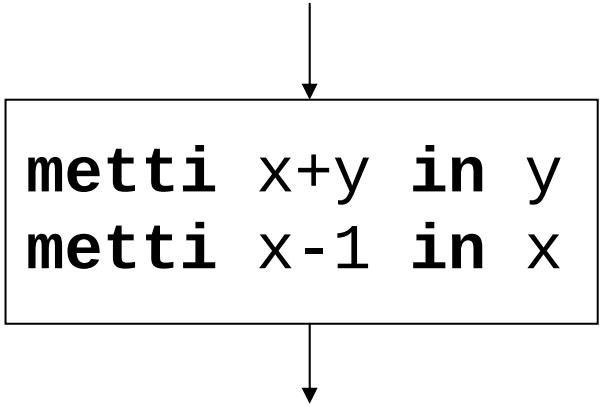
- Selezione
  - if, case, ...
- Iterazione
  - while, repeat
  - for

Si basano sull'uso di espressioni booleane

- Es. se x > 0, calcola la radice quadrata di x

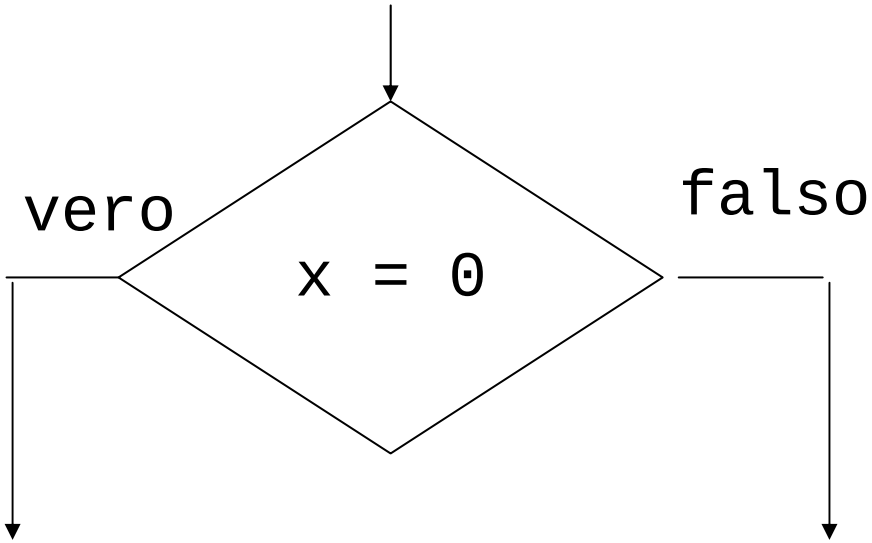
Diagrammi di flusso

- ***Blocchi di elaborazione***: contengono sequenze di azioni



Diagrammi di flusso

- ***Blocchi decisionali***: contengono una condizione booleana; se vera, si segue la freccia vero, se falsa si segue la freccia falso.

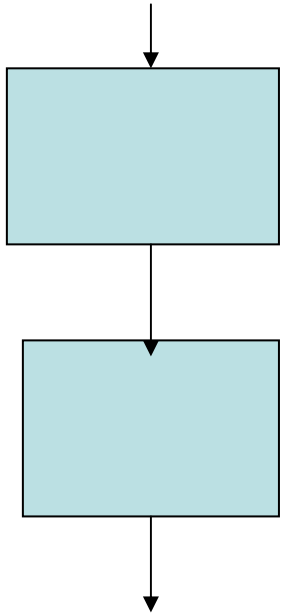


Diagrammi di flusso

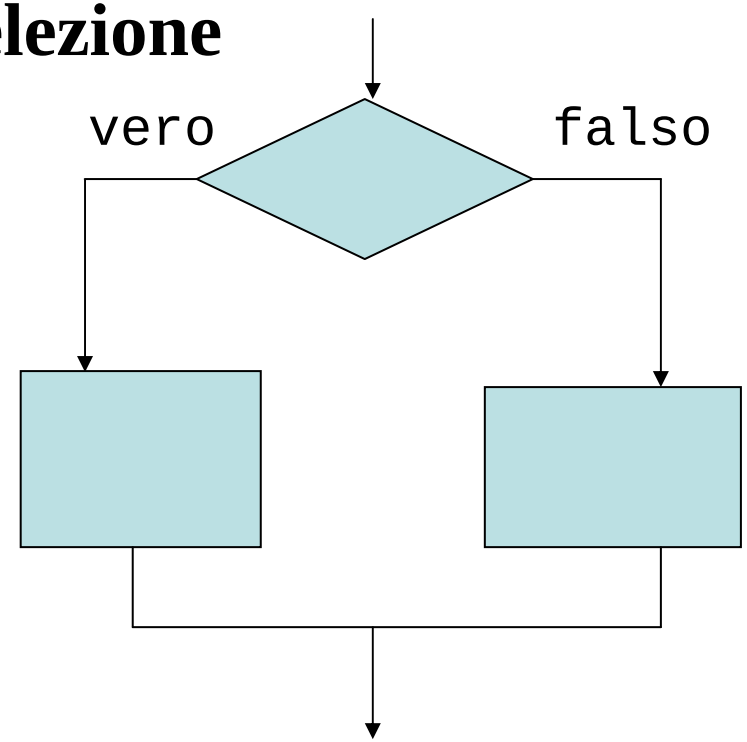
- Un diagramma di flusso si ottiene collegando le frecce uscenti dai blocchi di elaborazione e decisionali.

Strutture di controllo principali

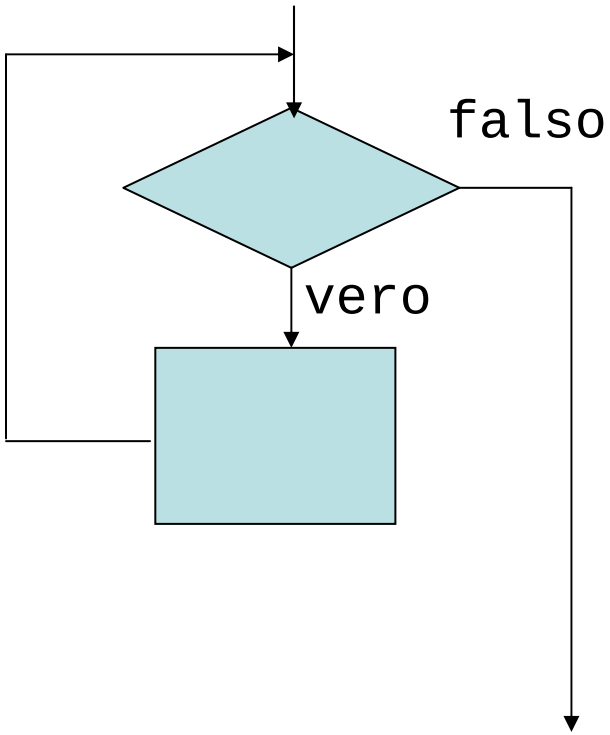
sequenza



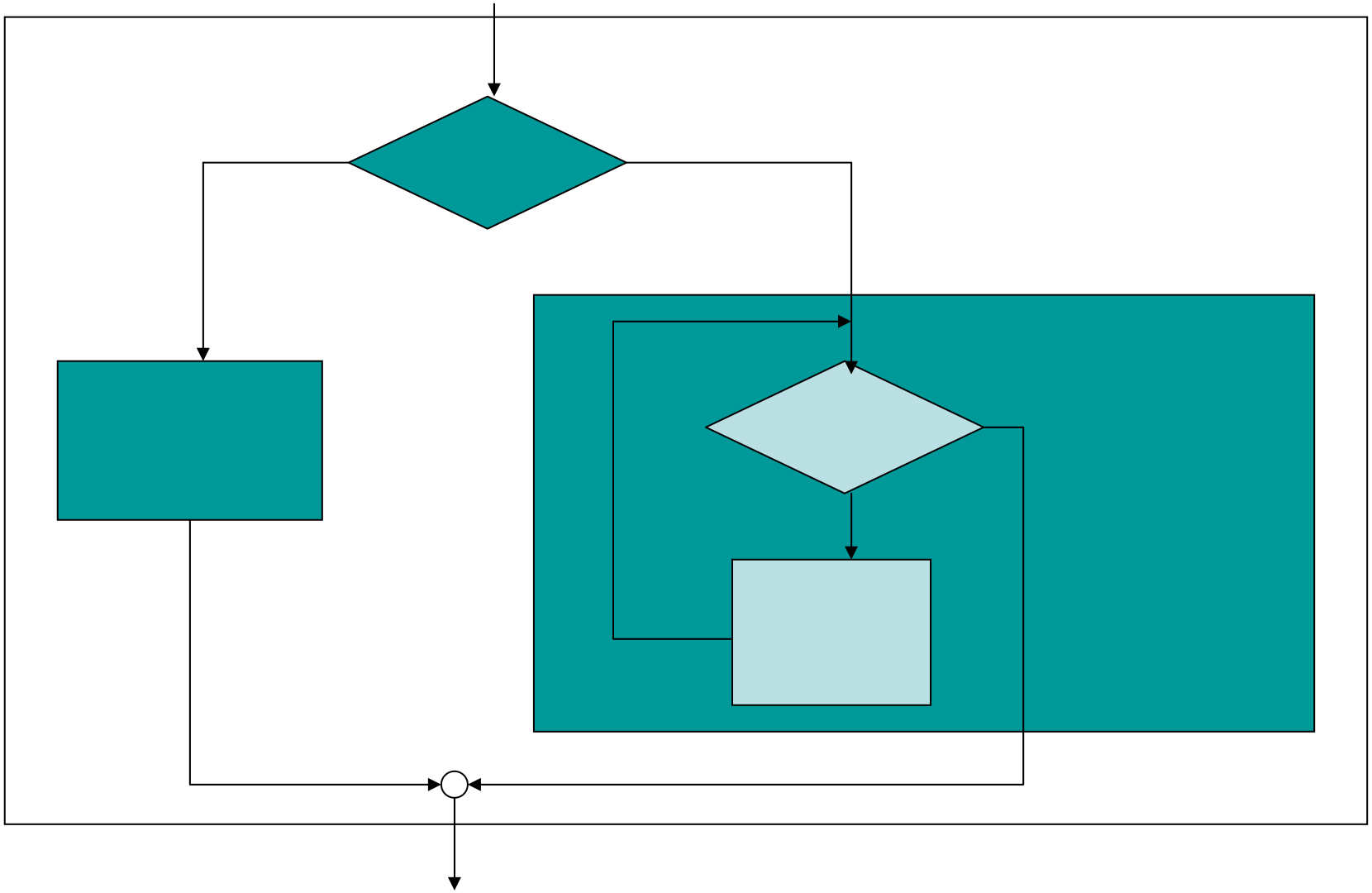
selezione



iterazione



Esempio di decomposizione modulare

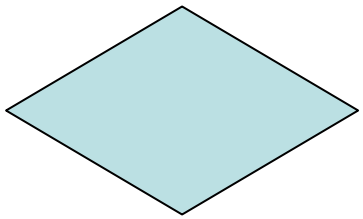


Simboli principali

- inizio o fine

- elaborazione

- decisione



- connessione

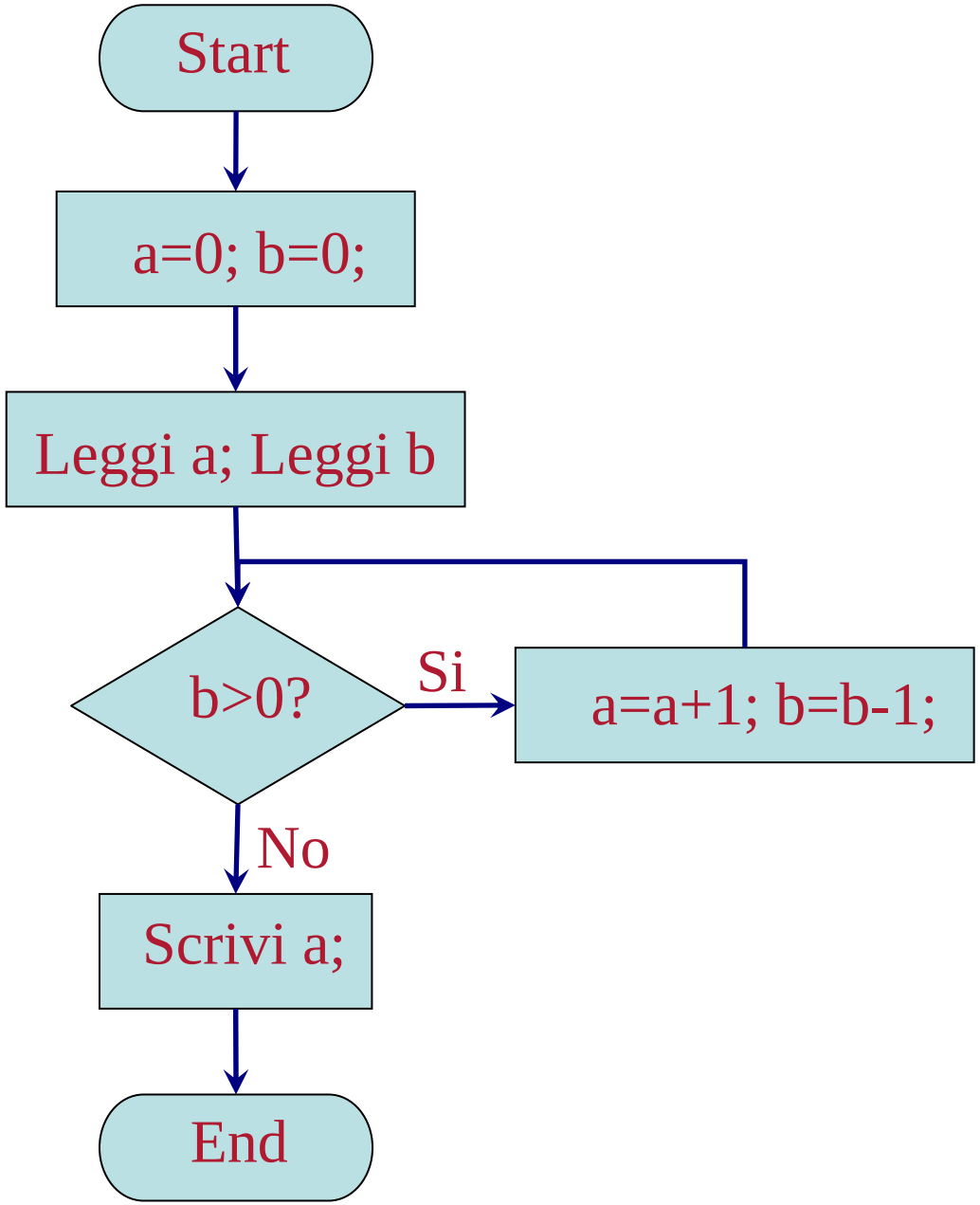
ESEMPIO:

Programma che legge due numeri a e b e calcola la loro somma, usando un linguaggio con le istruzioni:

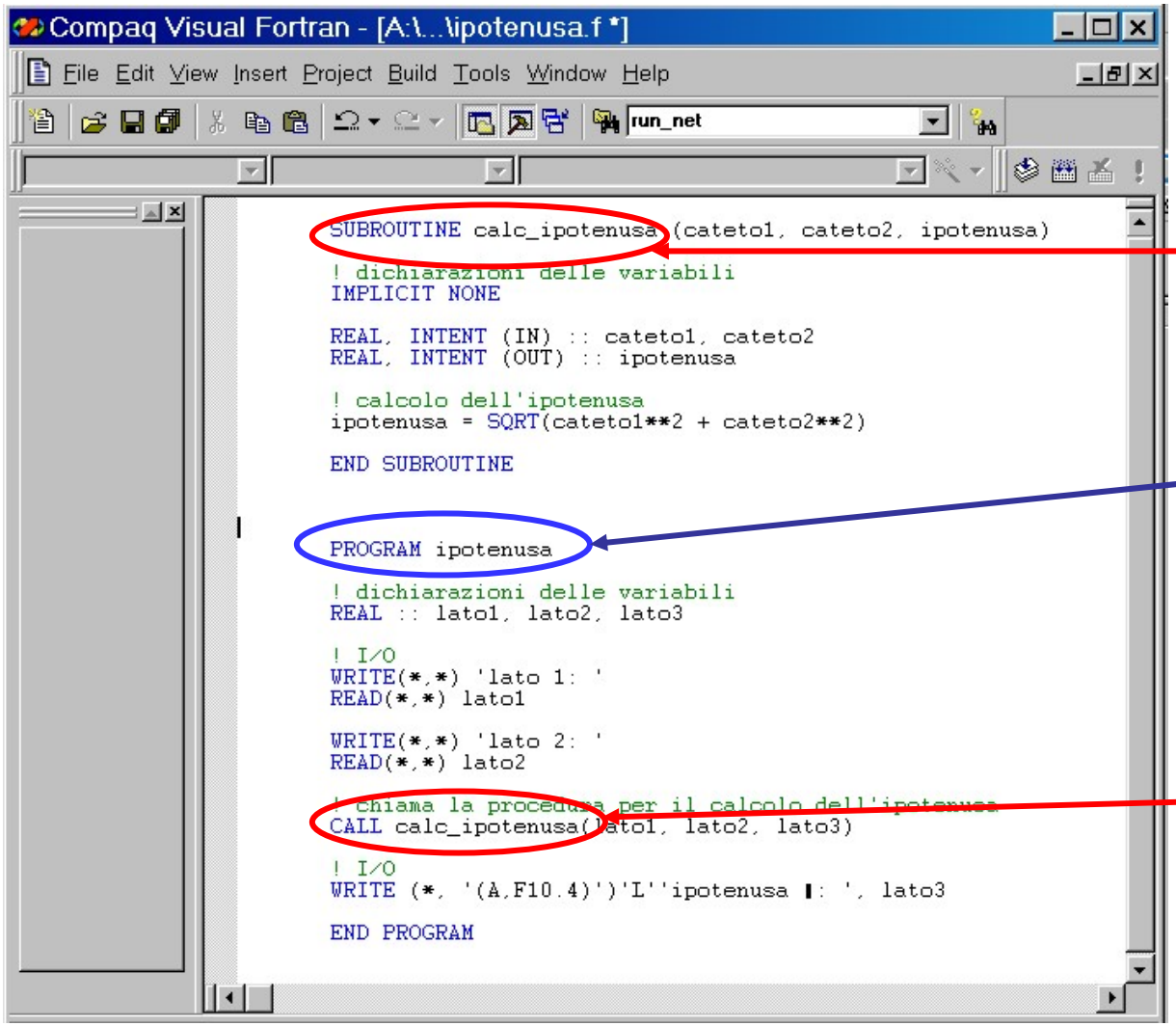
-Lettura

-Scrittura

-Somma\sottrazione di una unità per volta



Esempio di programma strutturato



procedura

programma principale

chiamata di procedura