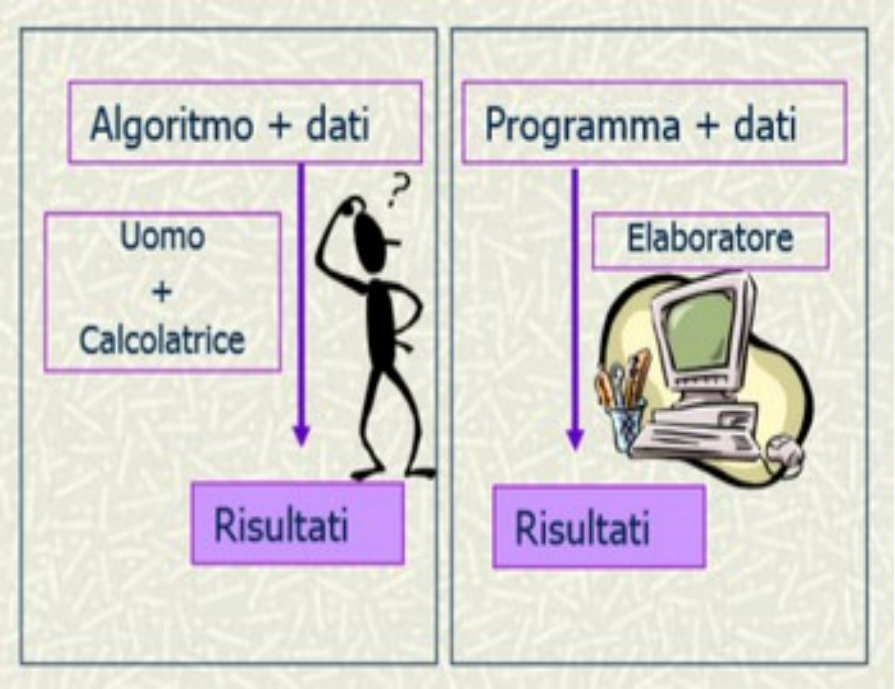
DIAGRMMI DI FLUSSO



Come scrivere bene codice? (1 di 1)

- Prima di iniziare a scrivere un programma:
  - Acquisire profonda comprensione del problema;
  - Progettare un approccio per la risoluzione del problema.
- Quando si scrive un programma è importante:
  - Sapere quali “building blocks” sono disponibili;
  - Usare tecniche di buona programmazione.

Algoritmi/Strutture (1 di 1)

- Algoritmo
  - Procedura che determina:
    - Azioni che devono essere eseguite;
    - Ordine in cui eseguire queste azioni.
- Strutture di Controllo
  - Specifica l'ordine in cui sono eseguiti statements.

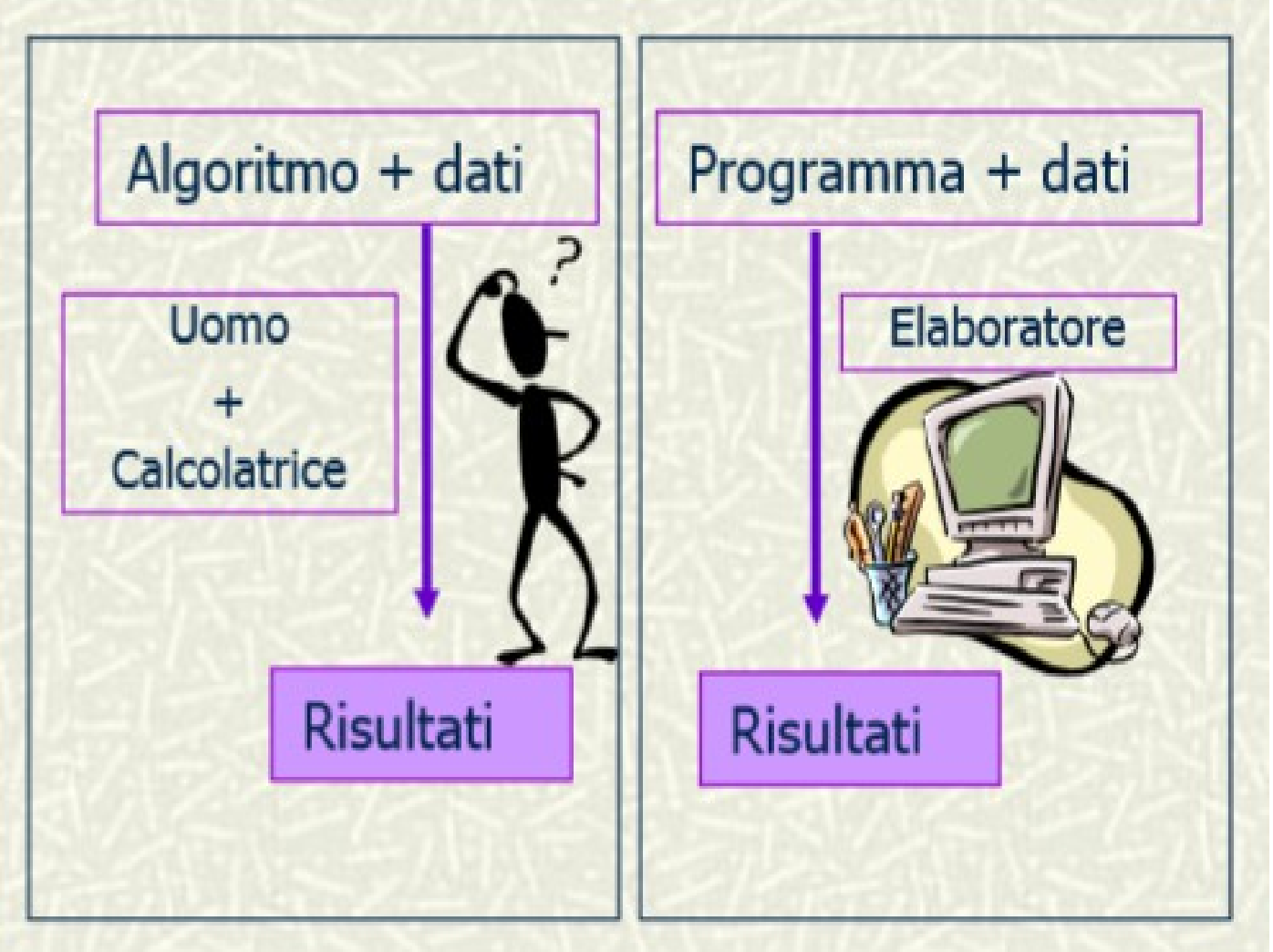
Algoritmi (1 di 3)

- Tutti i problemi di calcolo possono essere risolti eseguendo una serie di azioni primitive in un ordine specifico.

Algoritmi (2 di 3)

- Problema:
  - Effettuare un accredito su un c/c bancario.
- Soluzione :
  - Utilizzare un programma che serva per predisporre il calcolatore all'accredito di una qualunque cifra su un qualunque c/c.
- Note :
  - Cifra da accreditare e numero di c/c sono i dati caratteristici del problema.

Algoritmi (3 di 3)



Pseudocodice (1 di 1)

- Linguaggio informale per esprimere algoritmi;
- Simile (in qualche modo) al linguaggio naturale;
- Non è eseguibile su computer;
- “Pensare” come risolvere i problemi prima di scrivere codice;
- Facile convertire pseudocodice nel programma PHP corrispondente;

Strutture di Controllo (1 di 3)

- Esecuzione sequenziale:
  - Statement eseguiti uno dopo l'altro nell'ordine scritto.
- Trasferimento di controllo:
  - Statement da eseguire non è quello in sequenza.

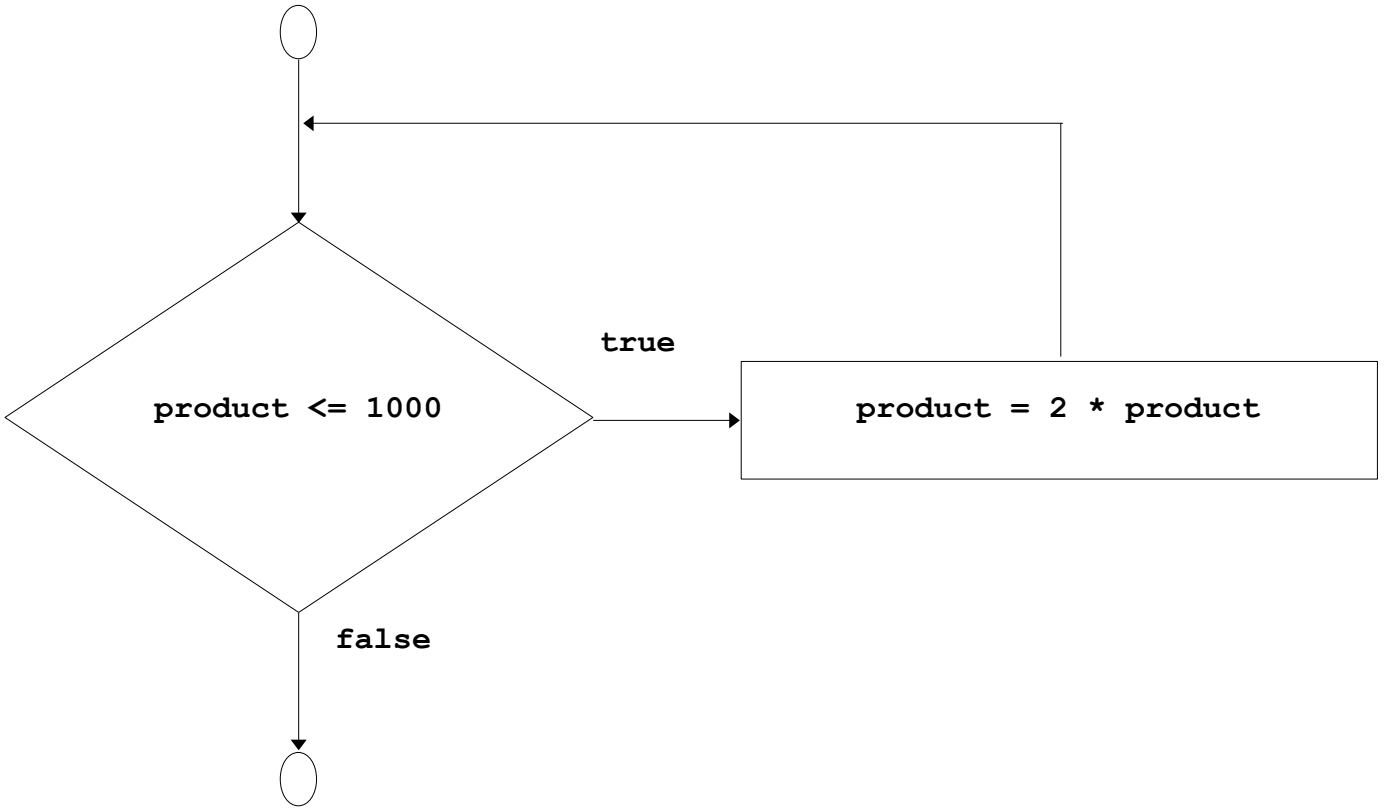
Strutture di Controllo (2 di 3)

- Teorema di Böhm-Jacopini:
  - Tutti i programmi possono essere scritti usando solo tre strutture di controllo:
    - Sequenziale;
    - Selezione;
    - Ciclo.

Strutture di Controllo (3 di 3)

- Sequenziale:
  - Built in, programmi eseguiti sequenzialmente per default.
- Selezione:
  - PHP ha tre tipi : if, if/else, e switch.
- Ciclo:
  - PHP ha tre tipi : while, do/while, e for.

Diagrammi di flusso (1 di 4)



Diagrammi di flusso (2 di 4)

- Rappresentazione grafica di un algoritmo ottenuto usando simboli speciali connessi da frecce (flowlines):
  - Rettangolo (simbolo di azione);
  - Ovale (inizio/fine);
  - Struttura di controllo (single-entry/single-exit).

Diagrammi di flusso (3 di 4)

- Rettangolo (simbolo di azione) :
  - indica ogni tipo di azione.
- Ovale (inizio/fine):
  - indica inizio/fine di programma o di sezione di codice.

Diagrammi di flusso (4 di 4)

- Struttura di controllo (single-entry/single-exit):
  - Connette punto di uscita di una struttura di controllo con punto di ingresso della successiva (control-structure stacking);
  - Rende programmi facili da costruire.

Struttura Selezione - if (1 di 5)

- Scegliere tra varie alternative;
- Esempio pseudocodice :
  - If voto studente >= 60
  - Stampa “Promosso”

Struttura Selezione - if (2 di 5)

- Se la condizione è true:
  - Viene eseguito statement Stampa “Promosso” e programma continua con statement successivo.
- Se la condizione è false:
  - statement Stampa “Promosso” viene ignorato e programma continua con statement successivo.
- Indentazione rende programmi più leggibili.

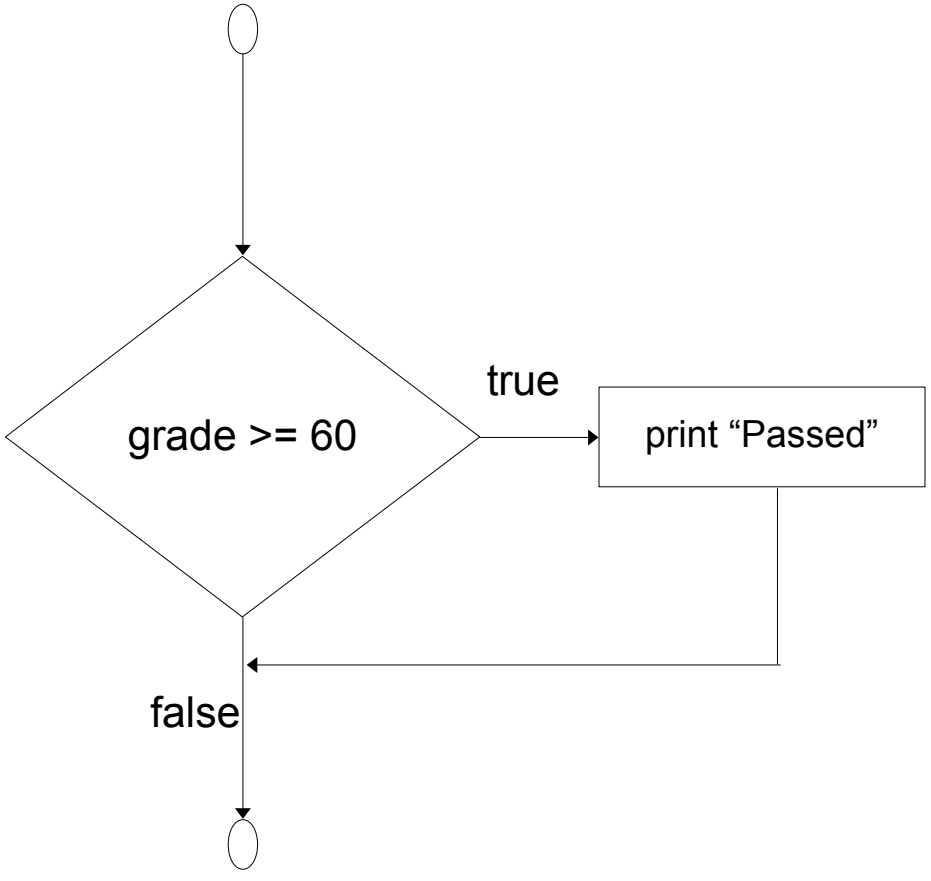
Struttura Selezione - if (3 di 5)

- Tradurre pseudocodice in PHP:
  - <?php
  - if ($grade >= 60) {
  - echo " Superato
"; }
  - ?>

Struttura Selezione - if (4 di 5)

- Tradurre pseudocodice in Diagrammi di flusso:
  - Rombo (decisione):
    - indica che bisogna prendere una decisione;
    - Contiene un espressione che può essere true o false:
      - Testa la condizione, segui percorso appropriato.
  - if single-entry/single-exit;

Struttura Selezione - if (5 di 5)



La decisione può essere presa su ogni espressione.

- Zero = >**`false`**
- Nonzero => **`true`**

Esempio:

`3 - 4` è **`true`**

Struttura Selezione – if/else (1 di 7)

- If:
  - Esegue un’azione solo se la condizione è verificata.
- if/else:
  - Si esegue un’azione diversa a seconda che la condizione sia verificata o meno.

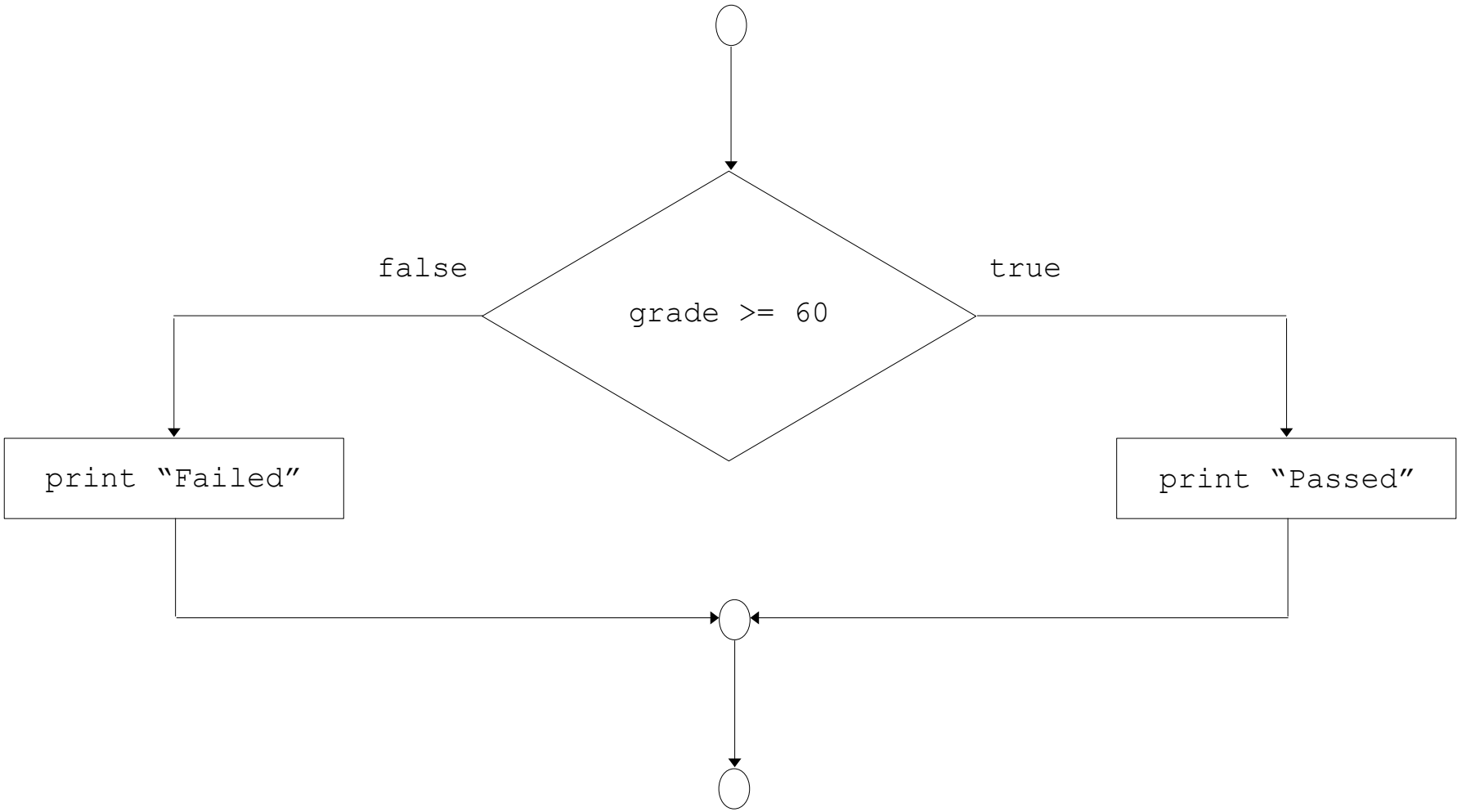
Struttura Selezione – if/else (2 di 7)

- Pseudocodice:
  - If il voto dello studente è più grande o uguale 60
  - Stampa “Superato”
  - else
  - Stampa “Non Superato”

Struttura Selezione – if/else (3 di 7)

- Codice PHP:
  - <?php
  - if ($grade >= 60) {
  - echo " Superato
"; }
  - else {
  - echo " Non Superato
"; }
  - ?>

Struttura Selezione – if/else (4 di 7)



Struttura Selezione – if/else (5 di 7)

- Operatore ternario condizionale (?:):
  - Ha tre argomenti (condizione, valore se true, valore se false).
- Pseudocodice potrebbe essere codificato come:
  - echo ($grade >= 60 ? “Superato” : “Non Superato”);

Struttura Selezione – if/else (6 di 7)

- Errori di sintassi:
  - Segnalati dal compilatore.
- Errori logici:
  - Errori che si presentano in esecuzione.
    - Errori logici “non fatali”.
      - programma va in esecuzione ma produce output non corretto.
    - Errori logici “fatali”.
      - programma finisce prematuramente.

Struttura Selezione – if/else (7 di 7)

- Errori di sintassi:
  - Esempio: if (x == 0) y = 0; esle y = 1;
- Errori logici.
  - Errori logici “non fatali”:
    - Esempio: if (x = 0) y = 0;
  - Errori logici “fatali”.
    - Esempio: x = 0; y = y/x;

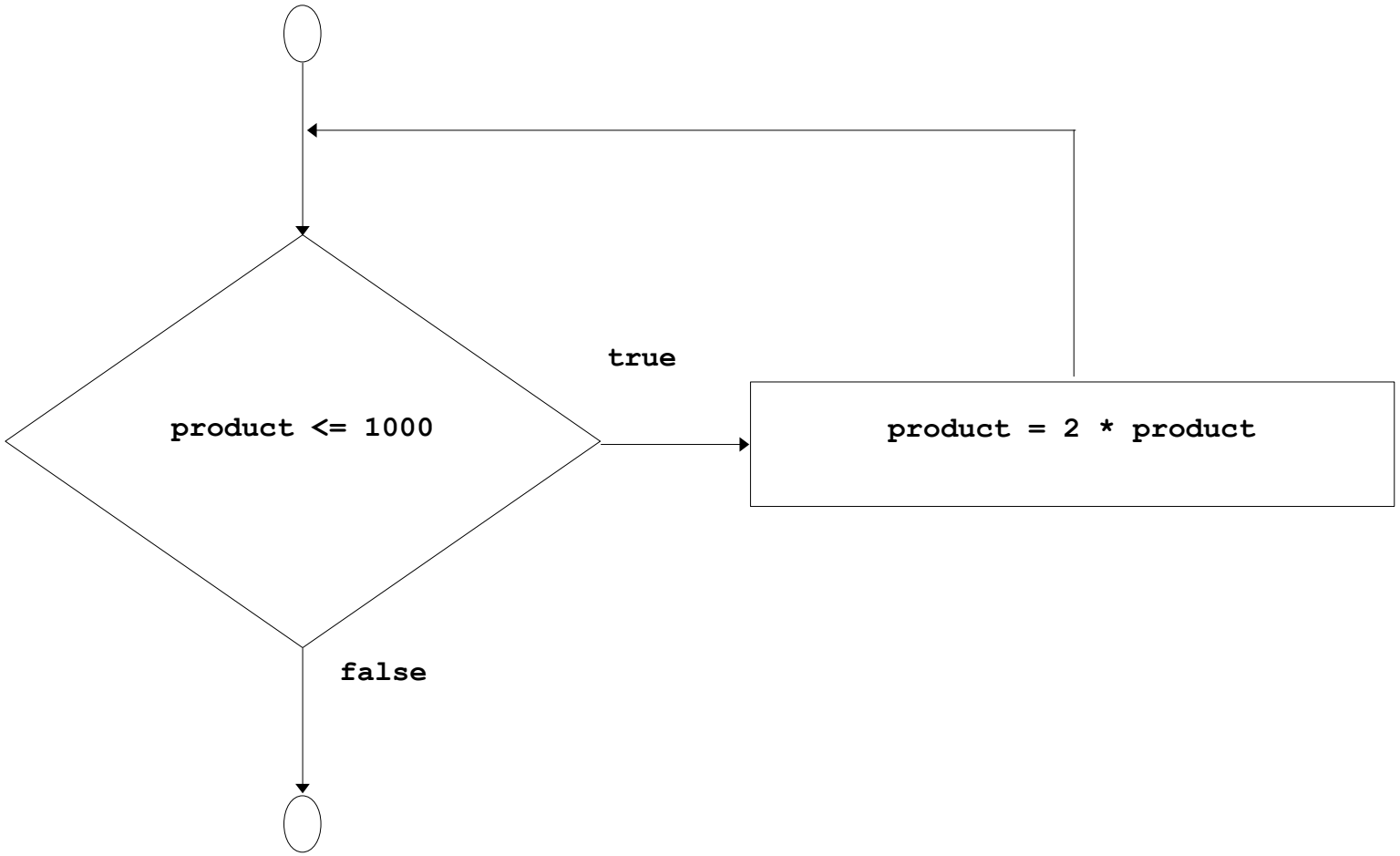
Struttura Selezione – while (1 di 3)

- Strutture di ciclo:
  - Si vuole specificare un azione che deve essere ripetuta fintantoché una determinata condizione è verificata;
  - Pseudocodice:
    - while ci sono più voci sulla mia lista della spesa
    - Acquista elemento successivo e scorri la mia lista
  - Ciclo while ripetuto fino a che la condizione non diventa falsa.

Struttura Selezione – while (2 di 3)

- Esempio
  - <?php
  - $product = 2;
  - while ($product <= 1000) {
  - $product = 2 * $product;
  - echo $product }
  - ?>

Struttura Selezione – while (3 di 3)



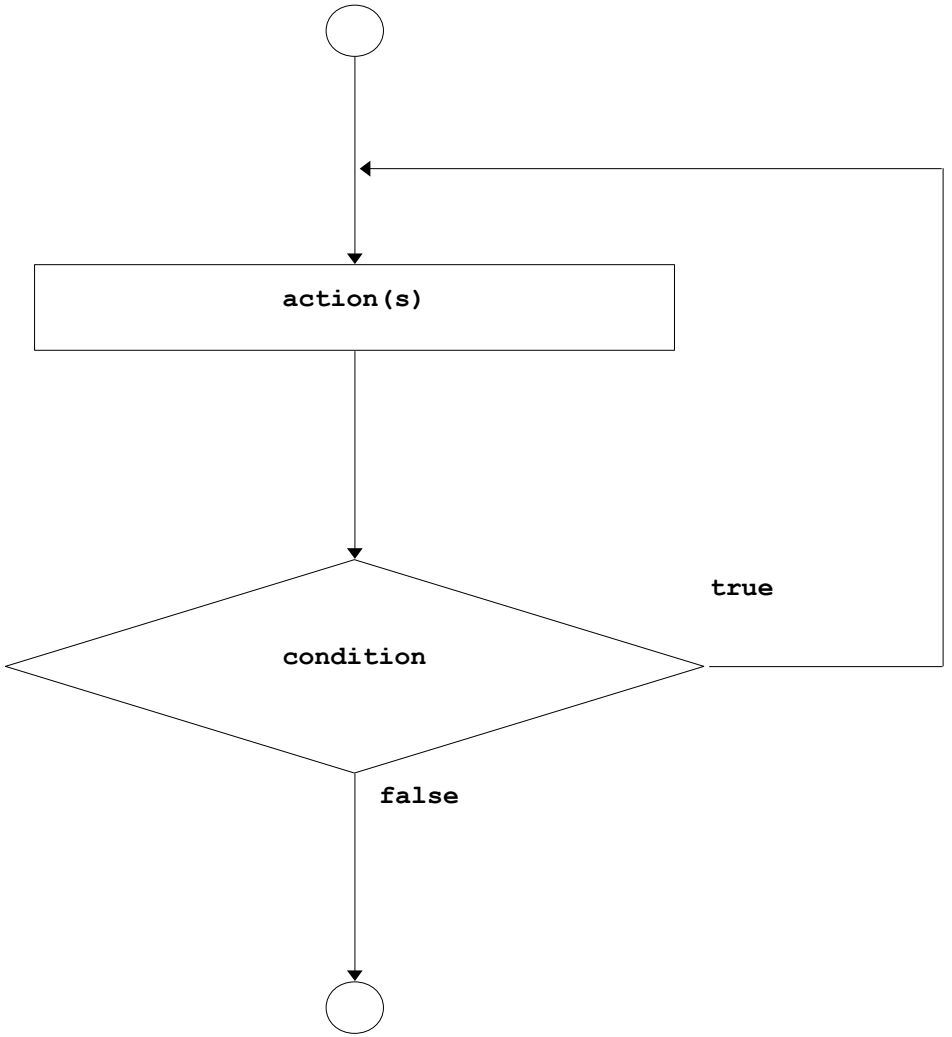
Struttura Selezione – do/while (1 di 3)

- Ciclo do/while simile al while:
  - Differenza: condizione di permanenza nel ciclo testata dopo l'esecuzione del corpo del ciclo.
- Formato:
  - do {
  - statement;
  - } while (condition);

Struttura Selezione – do/while (2 di 3)

- Esempio (prima $counter = 1;):
  - do {
  - echo $counter "\r ";
  - $counter++;
  - } while ($counter <= 10);
  - //Stampa gli interi da 1 a 10
- Tutte le azioni all'interno del ciclo vengono eseguite almeno una volta.

Struttura Selezione – do/while (3 di 3)



Progettazione top-down (1 di 10)

- La capacità di determinare la procedura di risoluzione di un problema è prerogativa dell'uomo.

- Il ragionamento che ci porta ad individuare la procedura di risoluzione si indica con il termine di **Analisi**.

Una tecnica di Analisi che si è rivelata molto valida è la tecnica **top-down**

Progettazione top-down (2 di 10)

- E' la tecnica che consiste nel definire la strategia di risoluzione di un problema attraverso livelli di dettaglio sempre più precisi, scomponendo il problema generale in sottoproblemi particolari.

Progettazione top-down (3 di 10)

- Supponiamo che il problema sia:
  - Scrivere un programma che esegua la media di un numero arbitrario (non noto a priori) di voti;
  - Numero arbitrario di studenti.
- Come sappiamo che dobbiamo fermarci?

Progettazione top-down (4 di 10)

- Valore “sentinella”:
  - Indica “fine dei dati in ingresso”;
  - Ciclo deve terminare quando viene immessa la sentinella;
  - Valore della sentinella scelto in modo tale da non essere confuso con input regolare (come ad esempio -1 per questo problema).

Progettazione top-down (5 di 10)

- Raffinamento per passi successivi, in modo Top-down (divide et impera):
  - Comincia con pseudocodice ad alto livello:
    - Determinare la media della classe per il quiz.
  - Dividi in sottoproblemi più piccoli:
    - Inizializzare le variabili;
    - Input, somma e contare i voti quiz;
    - Calcolare e stampare la media della classe.

Progettazione top-down (6 di 10)

- Molti programmi possono essere divisi in tre fasi:
  - Inizializzazione:
    - Inizializza le variabili del programma.
  - Processing:
    - Leggi dati in ingresso e modifica variabili di programma.
  - Chiusura:
    - Calcola e stampa risultati finali.

Progettazione top-down (7 di 10)

- Ci aiuta nella definizione di raffinamenti top-down.
- Raffiniamo fase di inizializzazione da:
  - Inizializzare le variabili
- A:
  - Inizializzare totale a zero
  - Inizializzare il contatore a zero

Progettazione top-down (8 di 10)

- Ci aiuta nella definizione di raffinamenti top-down.
- Raffiniamo fase di inizializzazione da:
  - Inizializzare le variabili
- A:
  - Inizializzare totale a zero
  - Inizializzare il contatore a zero

Progettazione top-down (9 di 10)

- Raffiniamo:
  - Input, sommare e contare i voti quiz
- A:
  - Input il primo grado (forse la sentinella)
  - While l'utente non ha ancora inserito la sentinella
  - Aggiungi questo grado nel totale parziale
  - Aggiungere uno al contatore grado
  - Input il grado successivo (forse la sentinella)

Progettazione top-down (10 di 10)

- Raffiniamo:
  - Calcolare e stampare la media della classe
- A:
  - If contatore non uguale a zero
  - Set la media al totale diviso dal contatore
  - Print “La media”
  - Else
  - Print “Nessun grado è stato inserito”