# Programmazione 1

# Lezione 1 - 27/09/2021

Informatica 🡪 Informazione più automatica

Computer 🡪 Supporto programmabile per la rappresentazione

Insieme dei processi che rendono disponibile:

* Elaborare e trasformare l’informazione
* Memorizzare l’informazione
* Distribuire l’informazione

Rappresentazione dell’informazione su un supporto materiale, il tutto va fatto secondo un’opportuna codifica (linguaggio macchina, codice ASCII)

## Codifica dell’informazione

Dobbiamo associare informazioni che vogliamo utilizzare in bit, ad esempio se vogliamo tradurre “ciao” in binario traduciamo ogni singola lettera tramite opportuna codifica. Ma la stessa informazione può essere rappresentata in diverse sequenze di bit (codifiche diverse).

## Il calcolatore

E’ una macchina che immagazzina e elabora dati, utilizza componenti elettronici e utilizza segnali digitali (Sistema binario 0/1, spento/acceso, falso/vero)

Il computer è capace di pensare quanto un sottomarino è capace di nuotare.

È quindi un supporto attivo per rappresentare ed elaborare le informazioni, può farlo in modi diversi (es. a seconda della fascia d’età, a seconda dei nostri gusti oppure a seconda delle proprietà degli oggetti).

Differenza rispetto ad altri dispositivi elettronici: il calcolatore è programmabile.

Il pallottoliere ci fa fare addizioni, rappresentando le informazioni con le palline. Invece il calcolatore permette di rappresentare informazioni più varie (mostrare immagini, impaginare un testo) e per fare ciò sono necessari sviluppare algoritmi o una procedura per eseguire le informazioni di nostro interesse. Ogni rappresentazione e informazione è costituita da una sequenza di 0 e 1, perciò sempre in binario. S

Si dice quindi che il calcolatore è “general purpose”, può elaborare qualsiasi tipo di informazione.

Per far fare nuovi compiti al calcolatore è necessario sviluppare un algoritmo e permettere al calcolatore di capire l’algoritmo. Il calcolatore decodifica il programma e lo esegue

## Architettura di Von Neumann

Prima proposta di architettura di un elaboratore, risale al 1946.

* Unità di elaborazione o CPU (Central processing unit)

1. Acquisisce e interpreta informazioni

* Memoria centrale

1. Contiene istruzioni e dati

* Periferiche: memoria di massa, input, output

1. Permettono scambio di informazioni con l’esterno

* Bus di sistema

1. Collega i vari elementi dal calcolatore

## Memoria

È una tabella, e in ogni cella è presente un bit, ogni riga si chiama “parola”. La prima colonna si chiama “indirizzo”, numera le colonne, ci permette di sapere in quale riga della colonna stiamo scrivendo/leggendo.

## Turing tumble

È un calcolatore di tipo meccanico

[www.turingtumble.com](http://www.turingtumble.com)

# Lezione 2 - 28/09/2021

## Introduzione a Java

## Astrazione

Si intende la distinzione tra proprietà esterne di un’entità e i dettagli di una struttura interna, sappiamo come usare un’automobile senza sapere com’è fatta dettagliatamente.

È presente una gerarchia di livelli, ogni livello è descritto tramite in suoi componenti. Conosciamo o il livello esterno, oppure i livelli più dettagliati.

## Algoritmo

Un algoritmo è una sequenza di istruzioni non ambigue eseguibili in modo automatico. Esempio: istruzioni per il lego, ricetta culinaria.

Esempio di ambiguità: fare soffriggere nell’olio delle cipolle (non sono definite le quantità).

L’algoritmo deve essere comprensibile, deve essere comprensibile a chi legge o a chi esegue il programma, se sarà un umano o un calcolatore.

Algoritmo + astrazione = divide et impera 🡪 Se ho un grande problema più difficile lo divido in sottoproblemi che uniti porteranno alla soluzione del problema più grande. Divido il programma in vari *moduli*. Scriverò il codice dei vari moduli e il codice che permetterà di interagire tra di loro.

## Proprietà degli algoritmi

* Correttezza: l’algoritmo funziona
* Efficienza: arrivo alla soluzione in modo rapido e usando la minor quantità di risorse (tempo e memoria)

## Algoritmi e programmi

Un computer è un esecutore di algoritmi descritti tramite programmi, il programma è una sequenza di istruzioni scritte in linguaggio *comprensibile* al calcolatore. L’algoritmo e il programma sono due cose diverse, il programma è la rappresentazione dell’algoritmo.

## Linguaggi di programmazione

Sono dei veri e propri linguaggi, sono dotati di sintassi e semantica.

* Primitive: sono le componenti base, i mattoni del linguaggio
* Sintassi: come è fatta una primitiva
* Semantica: il significato della primitiva

Esempio: La parola *Cane* - Sintassi: i 4 caratteri, Semantica: l’animale

**Regole**: spiegano come combinare primitive per rappresentare idee complesse

## Linguaggio macchina

Sono l’insieme delle istruzioni macchina, cioè quelle accettate dalla CPU (Codice binario)

Esempio: 16 bit per istruzione 🡪 4 bit per l’istruzione e 12 bit per gli operandi. Codice 0001 🡪 Semantica: Load (Preleva dalla memoria e scrive nel registro 0); Codice 0011 🡪 Semantica: Store (deposita nella memoria il contenuto del registro 0)

## Esempi di linguaggio macchina

Algoritmo: sommare il contenuto dei registri 5 e 6 e mettere il risultato nel registro 0

5056 🡪 0101 0000 0101 0110

Trasferire il risultato nella cella di memoria con indirizzo 6E

306E 🡪 0011 0000 0110 1110

F000 🡪 1111 0000 0000 0000

Il programma si trova in memoria e ha un indirizzo

Il linguaggio assembly è un linguaggio più comprensibile

## Linguaggi ad alto livello

Le primitive sono indipendenti dalla macchina, ogni processore può avere il suo linguaggio. Le primitive ad alto livello possono corrispondere a più istruzioni del linguaggio macchina.

Esempio: *costo = prezzo + spese\_sped;* Sto facendo più operazioni insieme, assegno un valore e faccio una somma.

Per tradurre il programma in linguaggio macchina:

compilatore 🡪 Traduce il linguaggio da alto livello a linguaggio macchina (traduco una sola volta e poi eseguo il file salvato)

Interprete 🡪 Traduce un’istruzione alla volta e, senza memorizzare il risultato, la esegue (devo tradurre ogni volta che interpreto prima di eseguirlo)

Il programma è indipendente dalla macchina, posso utilizzarlo su diverse macchine utilizzando l’opportuno compilatore.

Il linguaggio è standard, le specifiche sono chiare e universali (indipendentemente dall’hardware e dal sistema operativo), le estensioni del linguaggio sono aggiunte allo standard per aumentare le potenzialità

## Dal problema al programma

Quando vogliamo risolvere un problema 🡪 Troviamo un algoritmo 🡪 Scriveremo l’algoritmo in un linguaggio ad alto livello

Esempio: calcolare la somma dei due numeri

Algoritmo:

1. Legge il primo numero da tastiera

2. legge il secondo numero

3. somma i due numeri

4. Mostra a video il risultato

## Java Byte-code

Java cerca di unire i pregi dei linguaggi compilati e linguaggi interpretati. Java non traduce in linguaggio macchina ma in Byte-Code per un processore di una macchina virtuale, la JVM (java virtual machine). È simile al linguaggio macchina di tutti i processori ma non è direttamente eseguibile dal processore, ha bisogno di una fase successiva di interpretazione da un apposito interprete.

Vantaggi:

* Portabilità: codice di basso livello simile al codice macchina ma indipendente
* Velocità: tradurre ed eseguire byte-code è più veloce

La JVM traduce il byte-code in linguaggio macchina in base all’hardware.

Programma java 🡪 Compilatore java (javac nomefile.java) 🡪 programma in byte-code da interpretare (nomefile.class) 🡪 per eseguirlo (java nomefile (senza estensione .class!!!)) 🡪 Istruzioni tradotte in linguaggio macchina e eseguite

Per scrivere programma java:

* **Algoritmo**
* **Programma**
* **Compilare**
* **Interpretare**

## Tipi di errori

I linguaggi sono caratterizzati da sintassi e semantica.

Tipi di errori:

* **Sintattici:** Errori grammaticali che vengono intercettati e segnalati dal compilatore (sono i più facili da correggere)
* **Run-time**: i messaggi di errore non sono sempre d’aiuto, per esempio una *divisione per zero.*
* **Logici**: i più difficili da correggere e individuare, sono errori che possono esserci anche senza messaggi d’errore. Può essere un errore nel disegno dell’algoritmo, azioni sbagliate durante l’esecuzione. Dobbiamo testare il programma con dati opportuni che trattano le varie casistiche, dobbiamo predire il risultato prima dell’esecuzione. (es. mediaCinqueValori = sommaValori / 2 ; dovevo dividere per 5 e non per 2 per fare la media di cinque valori)

## Codifica informazione

Qualsiasi informazione deve essere tradotta in zero e uno per essere compresa dal computer.

Con un bit posso rappresentare due informazioni, con più bit posso rappresentare più informazioni.

8 bit = 1 byte: 28 configurazioni

Word (parola): dimensioni di una cella della memoria centrale (32 o 64 bit)

KB = 1024 byte

MB = 1024 Kbyte

GB = 1024 MByte

Determinare x (numero di bit) tale che 2x = M

Es. M = 50 🡪 X = 6

Infatti 25 = 32 (insufficiente)

Mentre 26 = 64 (sufficiente)

## Sistemi di numerazione

Comunemente usiamo la numerazione araba:

base decimale: dieci cifre

## Sistema binario

P = 2 Cifre 0 e 1

10110012 = 8910

## Codifica di caratteri

Ad ogni carattere corrisponde una sequenza di bit. Primo standard è stato il codice ASCII a 7 bit (128 caratteri) o a 8 bit (256 caratteri, esteso)

Per ciascuna lettera corrisponde un codice binario univoco:

a 🡪 1100001; n 🡪 1101110 ecc..

Se ho un testo (sequenza di bit), spezzo la sequenza in 7/8 bit e a ciascuno vado a cercare il valore corrispondente.

## Unicode

Standard attuale, assegna un numero univoco a ogni carattere, indipendente dalla piattaforma, dall’applicazione e dalla lingua. Codifica i caratteri di tutti gli alfabeti. Lunghezza variabile da 1 a 4 byte

# Lezione 3 - 04/10/2021

Vedi lezione3.java nella cartella *Programmi*.

## Elementi lessicali

Identificatore: l’identificatore è qualcosa che definiamo noi univocamente:

* Una classe
* Una variabile
* Un metodo
* Un oggetto

L’identificatore non può iniziare con un numero, possono essere usati solo numeri, lettere e qualche carattere speciale (\_ e $), non possono essere usate parole chiave del linguaggio (void, main etc..), non possono contenere spazi. Due variabili non possono avere lo stesso nome, le variabili sono case-sensitive.

La classe deve iniziare con lettera maiuscola e se sono più parole devono iniziare con lettera maiuscola (Prova, MyClass ecc..). Il nome della classe principale del programma e il nome del file salvato devono essere identici.

## Le variabili

Una variabile ha un unico tipo e un ben determinato *scope* (visibilità). Prima di poter utilizzare una variabile va dichiarata. Posso dichiarare più variabili sulla stessa riga, separando i nomi con una virgola. Il nome della variabile deve rispettare le regole di prima. Il nome della variabile deve essere significativo e deve iniziare con una lettera minuscola ed evitare di utilizzare il $.

Il tipo della variabile definisce che variabile può contenere e la sua dimensione.

Esistono due categorie di dati:

* Primitivi (int e double ecc.), sono i più semplici e non scomponibili
* Non primitivi (Classi, interfacce e array)