**INFORMATICA**

“Il nuovo Golem. Come il computer cambia la nostra cultura” - Giuseppe O. Longo

*Golem : A Praga un Rabbino aveva creato una creatura informe con scritto “EMET” (verità i*

*ebraico). Ad un certo punto la creatura si rivolta e il Rabbino gli cancella la “E”, diventando così “MET” (morte).*

E’ necessario avere almeno due stati per avere delle informazioni.

Definizione di informatica di Dreyfus:

“Insieme degli strumenti teorici e pratici che hanno lo scopo di elaborare l’informazione. Il termine deriva dal francese “informatique”, contrazione di “informatique automatique”.

Definizione operativa:

ELABORAZIONE

AUTOMATICA

DELL’INFORMAZIONE

Elaborare : compiere del lavoro all’informazione

mp3 : modalità di registrazione

Il suono viene campionato, non c’è curva continua, sono una serie di punti uniti.

INFORMATICA

Viene buttata gran parte dell’informazione in modo da essere raccolta in piccoli spazi.

Automatica : ha i fini tali informazioni che operazioni ripetitive vengano automatizzate

Informazione : contenuto di un messaggio trasferito da un soggetto ad un altro

* necessita di un supporto fisico

Per comunicare e recepire necessita di una variazione di stato di tutti gli elementi.

La variazione del sistema è legata alla variazione degli elementi che lo compongono.

2 tipi di sistemi : ANALOGICO (stati infiniti), DIGITALE (stati limitati)

L’informazione digitale considera sistemi digitali costituiti da elementi che possono assumere solo due stati : SISTEMA BINARIO

ALGORITMI

COMPUTER

ELABORAZIONE

AUTOMATICA

DELL’INFORMAZIONE

SCIENZA E

INGEGNERIA

DEI COMPUTER

HARDWARE

SOFTWARE

**SISTEMA BINARIO**

**(Liebeniz)**

BIT : binary digit

quantità minima di informazione di un sistema binario

un bit può assumere solo due valori 0 e 1

Falso/Vero

opposti Chiuso/Aperto

Off/On

\*Qu Bit : computer quantistici

(quantum bit) con stati infiniti

Digitalizzazione : misura degli stati degli elementi

1 elem. 2 elem.

4 stati

0 0

0 1

1 0

1 1

Per capire la quantità di stati dati in un tot di bit

**n**

2

In caso si ha un log2 5 e si ha un risultato non preciso si approssima sempre per eccesso.

Non tutti gli stati vengono occupati, solo quelli necessari, ma occupano comunque spazio.

**2** è il numero di stati

**n** è il numero di bit

inverso

**2**

IMMAGINE DIGITALE

è una matrice di punti (pixel)

ciascuno con un proprio colore o tonalità di grigio. Ha un numero definito di stati

**log x = n**

BYTE : insieme di 8bit

Il bit (informazione) vivendo nel software non è tangibile (no virtuale).

QR code è una rappresentazione del bit

nero : 1

bianco : 0

Ogni colore ha una precisa combinazione di bit, l’immagine anche se bidimensionale ha una sua profondità a livello di informazione, profondità colore.

Grigi : 8 bit

Colori : 24 bit (RGB 3 x 8)

Sistemi fino a base 4 sono innati, quindi il sistema decimale è artificiale.

**CAMBIO BASE**

da 25 (decimale) a sistema binario

25 : 2 = 12 + 1

12 : 2 = 6 + 0

6 : 2 = 3 + 0

3 : 2 = 1 + 1

1 : 2 = 0 + 1

25 (base 10) = 11001 (base 2)

algoritmo inverso

x

1 1 0 0 1

24 23 22 21 20

16 8 4 2 1

——————————————

16 + 8 + 0 + 0 + 1

25

La notazione esadecimale usata principalmente per il fatto che ciascuna cifra esadecimale può rappresentare numeri di quattro cifre binarie (bit), e di conseguenza due cifre esadecimali possono rappresentare 8 bit, cioè un Byte.

STRINGA : sequenza alfanumerica (numeri, lettere, simboli)

scritte in un font monospazio (occupano tutti la stessa larghezza)

Nel linguaggio HTML non è presente lo “spazio”, viene tradotto con % e numeri in base esadecimale = %20

# 66 66 66

R G B

Il colore viene codificato in base esadecimale perché valendo 24bit e se ogni cifra esadecimale vale 4 bit con 6 cifreesadecimali si può descrivere il colore in maniera compatta.

**ALGORITMO**

= elenco FINITO, NON AMBIGUO, REALIZZABILE

Per elaborare l’informazione di partenza, è quindi necessaria un’informazione iniziale.

ideatore = Muhammad ibn Musd al-Khwarizmi ( 780 - 850 )

matematico persiano che in un trattato d’algebra introduce l’algoritmo

2 strade:

1) 21 + 22 + 23 + 24 + 25 + 26 + 27 + 28 + 29 + 210 = 2026 (16 operazioni)

2) 210 + ( 210 + 2 ) = 2026 (3 operazioni)

Vale con qualsiasi numero e servono sempre 3 operazioni.

**RAPPRESENTAZIONE NUMERI NEGATIVI**

= modulo e segno

Si stabilisce per convenzione che se il numero è positivo il primo numero è “0”, se è negativo il primo numero è “1”.

-6 = 1 0110

+6 = 0 0110

*La parte evidenziata è il segno,*

*il resto, ovvero il vero numero è il modulo.*

Il numero di stati diminuisce, dimezzano i numeri positivi, ma si aggiungono i numeri negativi.

es.

+15

0 31 stati perché lo zero è sia positivo che negativo.

-15

**CODIFICHE**

CODIFICA ASCII (US-ASCII) - 1961

L’informazione è codificata in modo univoco, non più in base alla marca.

Approvata da ISO ( International - Standard - Organization )

Usava 7 bit (128 stati) :

95 caratteri stampabili ( numeri, lettere minuscole, lettere maiuscole, segni )

E’ stata poi estesa a 8 bit, vennero aggiunti segni e caratteri europei.

Rimane comunque limitato perché esclude altri alfabeti (ebraico, greco, arabo, cirillico, cinese…), così ISO crea diverse codifiche, ma che non sono comunicanti fra loro.

Nasce la Codifica UNICODE, anche se non tutti hanno aderito, o non completamente.

UNICODE è un sistema di codifica che assegna un numero univoco ad ogni carattere usato per la scrittura di testi, in maniera indipendente dalla lingua, dalla piattaforma informatica e dal programma utilizzato.

L'Unicode nasce nel 1987 su iniziativa di Joe Becker di Xerox e Lee Collins e Mark Davis di Apple. L'obiettivo del trio è studiare la fattibilità di un sistema di codifica del testo che fosse universalmente accettato e utilizzabile. Nell'agosto 1988 Joe Becker pubblica una prima bozza di quello che lui stesso chiama Unicode, “un sistema di codifica dei caratteri testuali che sia internazionale e multilingua”. Ed è in questo quadro di riferimento che Becker spiega anche l'origine del nome, teso a suggerire l'idea di un sistema di codifica “unico, unificato e universale”.

**I TRE PRINCIPALI AUTORI PER LA CREAZIONE DEL COMPUTER E DELL’INFORMATICA**

Hilbert

Gödel = teorema di incompletezza

Turing = macchina

Hilbert nel 1928 rivela l’esistenza di tre problemi della matematica:

* problema della completezza ( risolto da Gödel )

Il teorema di completezza di Gödel afferma che un sistema deduttivo per la logica dei predicati del primo ordine è "completo" nel senso che le regole di deduzione permettono di dimostrare tutte le formule logicamente valide. Un altro aspetto dello stesso problema è il teorema di correttezza che afferma che tutte le formule dimostrabili sono logicamente valide. Questi due teoremi, se dimostrati entrambi, implicano che una formula è logicamente valida se e solo se è dimostrabile (cioè è conclusione di una deduzione formale).

* problema della coerenza ( risolto da Gödel )
* problema della indecidibilità ( risolto da Turing )

Il Teorema di Turing asserisce l’esistenza di problemi non definibili, per i quali cioè non esiste alcun algoritmo di dare una risposta in tempo finito su tutte le istanze del problema. La dimostrazione a questo risultato si deve ad Alan Turing, che lo provò in un articolo del 1937. Il Teorema di Turing è in un certo senso la “versione informatica” del teorema di indecidibilità di Gödel.

Un accenno di dimostrazione viene dato di seguito.  
Supponiamo di avere alcuni programmi che stampano sempre o "si" o "no" in output, qualsiasi sia il loro input. Il nostro problema sarà definito come segue:

Dato sotto forma di codice un programma P che stampa sempre "si" oppure "no" ed un input I, stabilire l'output P(I).

Mostreremo per assurdo che non esiste alcun algoritmo in grado di dare una risposta per tutte le possibili coppie (P,I) di programmi ed input. Supponiamo quindi per assurdo l'esistenza di un programma che, preso il codice di un altro programma P, ci dica se esso stampa "si" oppure "no" quando esso riceve un certo input I; l'input del nostro ipotetico programma è quindi una coppia (P,I), mentre l'output è P(I). Chiamiamo questo programma . Ora modifichiamo leggermente il programma  creando un nuovo programma  che prende in input il codice di un programma P e quindi calcola ; questo nuovo programma stabilisce cosa stampa P quando riceve come input il codice di sé stesso. A questo punto modifichiamo ancora leggermente  in modo che esso stampi "no" quando il programma analizzato stampa "si" e viceversa; chiamiamo tale programma N (non esistente); notiamo che questo è appunto un programma che stampa sempre o "si" o "no". A questo punto chiediamoci lecitamente:

Cosa stampa N quando riceve in input N?

In pratica stiamo dando N come input a sé stesso, chiedendogli dunque di dirci cosa farebbe sé stesso ricevendo sé stesso come input. Ora, se la copia di N che viene analizzata stampasse "si" quando riceve N in input, allora il programma N analizzatore, che ha ricevuto proprio N in input, stamperebbe "no"; ma questo è assurdo. Viceversa, se l'N analizzato stampasse "no", allora N stamperebbe "si"; ma anche questo è assurdo. Pertanto tale programma non può esistere.

La versione originale della dimostrazione di Turing è in realtà basata sul problema della fermata,detto anche problema dell'arresto; si dimostra cioè che non può esistere un programma che possa decidere se un qualsiasi programma si arresti o continui a procedere all'infinito.

Una dimostrazione più formale richiede alcune nozioni di informatica teorica, come la conoscenza dei linguaggi formali e delle Macchine di Turing.

* Congettura di Goldbach ( non dimostrabile )

In matematica, la congettura di Goldebach è uno dei più vecchi problemi irrisolti nella teoria dei numeri. Essa afferma che ogni numero pari maggiore di 2 può essere scritto come somma di due numeri primi (che possono essere anche uguali).

Per esempio :

4 = 2 + 2

6 = 3 + 3

8 = 3 + 5

10 = 3 + 7 = 5 + 5

**LINGUAGGIO FORMALE**

L’Informatica necessita di un linguaggio chiaro, un

LINGUAGGIO FORMALE = ARITMETICA (non equivocabile)

può essere usato come un “gioco”, gli scacchi, che incarnano il linguaggio del computer perché non creano ambiguità, mosse precise, risultati finiti.

Il computer ha sempre ragione perché fa uso di un linguaggio formale.

LINGUAGGIO FORMALE: è un formalismo e quindi necessita di una sintassi e di una semantica, definite in modo rigoroso (senza eccezioni).

Leibliz si è preoccupato prima di Hilbert del problema della dicibili, è convinto del fatto che il pensiero umano sia meccanizzatile (pensiero poi ripreso da Hilbert).

Turing dimostra l’impossibilità di questo pensiero tramite le regole di Frege (inizialmente usato della precedente teoria).

Tutti i software sono macchine limitate.

macchina universale = U

macchina limitata = L

**U**

L 1

L 2

L 3

L n

La macchina universale è in grado di imitare tutte le macchine limitate.

**ALGEBRA BOLEANA**

L’algebra di Bole, anche detta algebra boleana, è una variante dell’algebra elementare che opera

TRANSISTOR: come un normale interruttore, consente o meno il passaggio di corrente.

Sono attivati da un segnale elettronico binario, non genera un movimento principale.

C’è sempre un segnale

INTERRUTTORE NORMALE INTERRUTTORE NORMALMENTE CHIUSO

A = 0 A = 0

A = 1 A = 1

3 circuiti alla base del computer:

Circuito NOT : porta logica (operatore boleano), normalmente chiuso,

negazione del segnale di comando (inverte).

Circuito AND (in serie) : Porta logica (operatore boleano), normale, la corrente passa solo se A e B sono contemporaneamente uguali a 1.

Circuito OR (parallelo) : porta logica (operatore boleano)

Con questi circuiti si può costruire qualsiasi tipo di circuito elettrico digitale semplicemente combinandoli.

INGRESSI / IMPUT

CIRCUITO

ELETTRICO

USCITE / OUTPUT

Ogni circuito deve avere solo un’uscita.

E’ importante rispettare l’ordine degli ingressi.

TABELLE DELLA VERITA’

REGOLE PER L’ADDIZIONE BINARIA

0 + 0 = 0

0 + 1 = 1

1 + 0 = 0

1 + 1 = 0 con riporto di 1

**MONITOR**

Primo monitor : TUBO CATODICO (CRT), nasce negli anni ’40, va a fosfori

- bassa definizione

- pericolosi

- economici

- effetto memori (pixel danneggiati)

Monitor LCD (2004) sfruttano le proprietà ottiche dei cristalli liquidi contenuti all’interno di piccole cellette. L’immagine è costruita, come già negli schermi CRT, modulando l’intensità luminosa a gruppi di tre subpixel rossi, verdi, blu (RGB).

I cristalli liquidi originariamente sono trasparenti; viene aggiunta una retroilluminazione, ovvero neon posti ai lati che favoriscono la qualità e la quantità di colori; indipendenza cellette; effetto scia ( LCD - TFT subpixel indipendenti )

Monitor al PLASMA, PDP e al LED

Celle perfettamente sagomate, racchiuse fra due pannelli di vetro che contengono una miscela di gas nobili ( neon e xeno ).

Il neon e lo xeno vengono fatti agitare = fluido elettrico, che va contro le pareti ricoperte da uno strato di fosforo (effetto memoria).

Grande luminosità; ampio gamut di colori; nero di tipo “dark-room”; dimensione fino a 150 pollici; consumo di energia superiore rispetto al CRT e LCD

Unità I / O (input / output) = ibrida, ovvero consente di ricevere e dare informazioni

Memorie di massa: dispositivi di memorizzazione secondari o ausiliari, sono dispositivi

I/O, memorizzano grandi quantità di dati.

TIPOLOGIE:

* nastri magnetici (facilmente riparabili) I/O
* dischi magnetici I/O
* memorie di stato solido (es. USB) I/O
* CD ibridi solo inizialmente, una volta scritti non si possono più sovrascrivere
* DVD

monitor e stampante > analogici

unità

output

MONITOR : l’unità di misura delle dimensioni di un computer è il pollice (2,54 cm).

Per “dimensione del monitor” si intende la distanza misurata in diagonale.

RISOLUZIONE: non è la quantità, ma la dimensione di pixel

VELOCITA’ DI REFRESH (espressa in Hertz): velocità di ridisegnamento dell’immagine, minimo 75Hertz (comunque si nota)

DOT PITCH : misura la diagonale + spazio bianco del pixel e fra i pixel monitor di accettabile qualità = 0,28mm (diagonale)

**CHIP - CIRCUITI INTEGRATI**

AND e NOT = NEND (solo per i chip)

I primi computer usavano delle valvole termoioniche,

Nel 1958 primo chip (14 anni dopo il computer)

Memoria temporanea (RAM: random, access, memory) = i Registri

Circuito FLIP-FLOP, ognuno memorizza solo 1bit e funziona solo con la corrente è quindi considerato temporaneo.

Composto da 2 NOT, minimo per una cella da 1 bit, ogni cella ha 3 bit.

Funziona solo con la corrente, è quindi temporaneo.

Unità Logico-Aritmetica (ALU)

8 bit

Registi

8 bit

BUS : sistema che permette lo scambio di informazioni ordinato (microcircuiti) > problema di velocità ( si misura in Hertz).

Apre e chiude due porte in modo da far passare una informazione alla volta.

Il circuito che lo regola = controller.

4 operazioni

LATCH (porticina)

Istruzione

**ARCHITETTURA DI VON NEUMAN**

* 1944 : calcolatore elettronico ENIAC

“immagazzinare sia i dati che le istruzioni nella memoria del computer”

obbiettivi:

* distinguere le unità di elaborazione delle informazioni dalle unità di memorizzazione
* le componenti principali
* usare un BUS

UNITA’ CENTRALE

DI ELABORAZIONE

(CPU)

CONTROLLER

(unità di controllo)

ALU

(unità logico-aritmetica)

REGISTRI

UNITA’ DI

MEMORIA

UNITA’ DI

IMPUT

UNITA’ DI

OUTPUT

BUS

RAM : composta da 8 chip da 8 bit + 1 chip di parità (in grado di ricostruire un’informazione se un chip la perde)

PROCESSORE è in un ciclo continuo, viene interrotto da vari “interrupt”, nuove richieste.

**INTRODUZIONE AI SISTEMI OPERATIVI**

Il sistema operativo è quel software particolare che si interpone fra l’utente e l’hardware.

Il software ora è preinstallato, in passato si installavano tramite un disco.

utente

Software applicativo

Interfaccia grafica: scrivania dove giacciono i software (applicazioni)

Sistema Operativo

Software di sistema (Utility)

hardware

LINUX (1991)

è una variante di UNIX (1969)

(opensource)

ha un’interfaccia grafica

WINDOWS (1990)

numerose varianti nel passato

1981 : MS (Microsoft - Bill Gates) - DOS 1.0 per IBM

non ha interfaccia grafica

1990 : Primo sistema Windows 3.0

basato su MS - DOS

1993 : Windows NT (new technology) 3.1

Microsoft compra la DIGITAL

MAC OS 1984 (Lisa Os > comprende anche il mouse)

Creato da Steve Jobs e Steve (polacco)

1987 Macintosh 128 k, con mouse.

Inizialmente il mouse è stato rifiutato.

“Fin dalla sua fondazione Apple Computer ha avuto una missione: portare la potenza e la versatilità dei computer alle persone normali.”

XEROX inventa mouse e interfaccia grafica, Apple sfrutta queste invenzioni,

nesce così Lisa Os.

Tutti i sistemi operativi sono basati su UNIX, tranne Microsoft

**ALGORITMI E LINGUAGGI DI PROGRAMMAZIONE**

L’informatica deve avere una ricaduta reale.

**Strategia Polya : Capire il problema**

* Capire il problema.
* Individuare le incognite, i dati ed eventuali condizioni.
* Vedere se le condizioni possono essere soddisfatte.
* Come determinare il termine incognito.
* Vedere se ci sono condizioni ridondanti o informazioni contraddittorie.
* Disegnare la figura.
* Utilizzare la giusta notazione.

Scelta di una strategia appropriata e questa si affina risolvendo molti problemi. Bisogna quindi:

* trovare il collegamento tra i dati e l’incognita;
* utilizzare eventualmente problemi ausiliari;
* ottenere alla fine un programma della soluzione;
* chiedersi se il problema somiglia ad altri già risolti;
* comprendere quale teorema può essere utilizzato;
* interodurre, se necessari, elementi ausiliari;
* riformulare, eventualmente, il problema;
* risolvere, eventualmente, un problema correlato e più accessibile;
* risolvere parti del problema;
* controllare di aver utilizzato i dati a disposizione.

**DIAGRAMMI DI FLUSSO**

INIZIO

FINE

(rettangolo con angoli smussati)

BLOCCO LETTURA (es. prompt e array)

BLOCCO SCRITTURA (es. document\_write)

BLOCCO DI AZIONE (operazioni)

?

vero

BLOCCO DI CONTROLLO

falso

**SUBLIME TEXT**

Java Script ha bisogno del linguaggio HTML

Java Script esiste tra due parentesi graffe; nasce con i browser > Net-Scape comunication, per rendere dinamiche le pagine statiche.

(diverso da Java)

Deve essere interpretato (ha bisogno di un browser per essere letto, ma non in tutti funziona) è diverso da “compilato” (es. icone).

E’ possibile commentare lo script utilizzando slash //, ma in questo caso è possibile utilizzare solo una riga, per più righe /\*, da ripetere per chiudere il commento.

SPAZIO NON RICONOSCIUTO

IDENTAZIONE: esempio “<tile>”

carattere monospazio

ogni comando termina con “;” oppure il comando è incompleto (in alcuni casi non va messo)

var nome = 2 ;

c’è una nuova variabile

contenuto

nome variabile

*Sbagliato dal punto di vista logico, un nome non può rappresentare un numero*

**CARATTERI AMMESSI E NON, NEI NOMI DELLE VARIABILI**

NON AMMESSI

* numeri (se al primo posto)
* spazio
* segni di punteggiatura

AMMESSI

* numeri
* lettere alfabeto minuscole e maiuscole
* underscore e trattino

I numeri con la virgola mobile rallentano (occupano più spazio)

//STRINGA

definita dagli apici “”, ‘’, JavaScript identifica la stringa

per spaziare nel nome bisogna utilizzare \_ , - , maiuscola

//Tipologia BOLEANA

== allora

se c’è “if” non serve ;

var : variabile locale

senza var : variabile globale

SWITCH

è sempre legato a un CASE

( ) : condizione della variabile

Case coerenti con la variabile che sto testando

**for** : è più compatto; usato quando si sa quante volte ciclare

**while** : quando non si sa quante volte ciclare

**array** : composto da più variabili

**ESEMPI**

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function test() {

// una sola riga di commento

/\* più righe di commento

per chiudere ripetere \*/

// VARIABILI

// come si definisce una variabile

/\* Caratteri ammessi nei nomi delle variabili

Non ammessi:

-spazio

-numeri (se al primo posto)

-segni di punteggiatura

Ammessi:

-lettere alfabetiche minuscole e MAIUSCOLE (non accentate)

-numeri

-underscore e trattino \*/

// Tipologia NUMERI

var numero = 2; // numero intero

var numero\_con\_i\_decimali = 3.14115; // numero in virgola mobile

// Tipologia STRING

var nome = "Mario";

var cognome = "Vabbè";

var altraStringa = 'Ciao'; /\* Per avere gli apici doppi ("") utilizzare apici semplici ('') e viceversa \*/

var altraStringaAncora = "Ho detto: 'Oggi non piove'";

var ultimaStringa = "Benvenuti all'Hdemia";

var eraQuestaLUltima = 'allora disse: "va bene così"';

var cosa\_e = "123456"; // è comunque una stringa

// Tipologia BOOLEAN

var finito = false; // inteso come "0"

var ancora = true; // inteso come "1"

//Tipologia NaN, Not a Number

//var risultato\_divisione = 5/0; // blocca lo script perchè non ha risultato

//Tipologia UNDEFIND

var indirizzo; // blocca lo script

//STRUTTURE DI CONTROLLO

/\*

var primo\_numero = 2;

if (primo\_numero == 2) //se c'è if non serve ;

{

document.write("primo\_numero e' uguale a 2"); // La verità della frase non viene decisa dal computer, funziona anche con 3

//struttura più semplice di if

}

/\*

if (condizione) {

Codice da eseguire

}

else {

Codice alternativo da eseguire

}

\*/

/\* primo\_numero = 1; //essendo già esistente la variabile il comando è valido

var secondo\_numero = 3;

if (primo\_numero > secondo\_numero) {

document.write(primo\_numero + "e' maggiore di" + secondo\_numero); /\* Per avere come risultato il numero al posto di "primo\_numero" e "secondo\_numero" stringare solo la frase e aggiungere i + per collegare il numero alla stringa.

}

else {

document.write(primo\_numero + "non e' maggiore di" + secondo\_numero);

}

/\*

if (condizione) {

Codice da eseguire

}

else if (condizione alternativa) {

Codice alternativo da eseguire

}

else {

Codice da eseguire in tutti gli altri casi

}

\*/

var primo\_numero = Number (prompt("Inserisci il primo numero")); /\* prompt è l'avviso all'utente ad inserire il numero con "Number e le parentesi tonde è possible rendere una Stringa Numeri" \*/

var secondo\_numero = Number (prompt("Inserisci il secondo numero"));

if (primo\_numero > secondo\_numero) {

document.write(primo\_numero + "e' maggiore di" + secondo\_numero);

}

else if (primo\_numero < secondo\_numero) /\* quanti ne voglio, ma l'ordine dev'essere mantenuto\*/ {

document.write(primo\_numero + "non e' maggiore di" + secondo\_numero);

}

else /\*solo uno, tenere ordine e non aggiungere spazi \*/ {

document.write(primo\_numero + "e' uguale a " + secondo\_numero)

}

/\* se vengono messe delle lettere al posto di numeri si ha NaN\*/

/\* COMPARATORI

== uguale

>= maggiore o uguale

<= minore o uguale

> maggiore

> minore

!= diverso

\*/

} // Chiude la graffa aperta dopo function

</script>

</head>

<body onload = "test();">

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function test() {

var giorno = 3; //può cambiare

switch (giorno) /\*ogni switch ha almeno un "case", tra uno e l'altro è necessario un "break", senza si può comportare in modo anomalo \*/

{

case 1:

document.write("lunedi'");

break;

case 2:

document.write("martedi'");

break;

case 3:

document.write("mercoledi'");

break;

case 4:

document.write("giovedi'");

break;

case 5:

document.write("venerdi'");

break;

case 6:

document.write("sabato");

break;

case 7:

document.write("domenica");

break;

default: //necessaria la frase di errore in case la casistica non sia presente

document.write("Il numero inserito non e' corretto")

break;

/\* non si può usare la struttura usata con if la volta precedente perchè bisognerebbe scrivere tutte le casistiche, quest'ultima è la differenza sostanziale tra if e switch\*/

}

var lingua = prompt("Digita la lingua");

switch (lingua)//graffa dopo la descrizione della struttura

{

// scrivere il case in base alla variabile trattata, in questo caso bisogna usare la stringa

case "italiano":

document.write("ciao");

break;

case "english":

document.write("hi");

break;

case "espanol":

document.write("hola");

break;

default:

document.write("I apologize. I don't know what you are asking me!");

break;

}

} // Chiude la graffa aperta dopo function

</script>

</head>

<body onload = "test();">

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function test() {

//STRUTTURE DI CICLO (LOOP O INTERAZIONE)

/\*c'è sempre una struttura di controllo, ma che è inglobata nella struttura di ciclo.

La prima è:

for (inizializzazione; controllo; incremento) {

Codice da eseguire ripetutamente

}

\*/

for (var i = 1; i <= 10; i = i + 1){

document.write(i);

document.write("<br>");

}

for (var i = 10; i >= 1; i = i - 1){

document.write(i);

document.write("<br>");

}

/\*

la seconda è:

while (condizione) {

codice da eseguire

}

\*/

i = 1;

while ( i <= 10 ) // la variabile è già deefinita in precedenza

{

document.write(i);

i = i + 1;

}

/\* il "for" è più compatto e si usa quando si sa quante volte ciclare, il "while" viene isato quando non si sa quante volte ciclare\*/

// ARRAY (liste, tabelle o vettori in italiano)

/\* è un contenitore di tipologie di variabile che usa le parentesi quadre []\*/

var elenco = ["gennaio","febbraio","marzo","aprile","maggio","giugno","luglio","agosto","settembre","ottobre","novembre","dicembre"];

document.write("<br>");

var mese = Number (prompt("Inserisci il numero del mese"));

if (( mese >= 1 ) && ( mese <= 12 )) {

document.write(elenco[mese - 1]);//in guesta riga la stringa viene letta come numero perchè gli si vuole far fare un'operazione

}

else {

document.write("Il numero del mese non e' compreso tra 1 e 12");

}

} // Chiude la graffa aperta dopo function

</script>

</head>

<body onload = "test();">

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>JavaScript</title>

<script>

function test() {

var giorno = ["lunedì","martedì","mercoledì", "giovedì","venerdì","sabato","domenica"]

var numero\_giorno = prompt("Inserisci il giorno della settimana");

if (numero\_giorno > 7) {

document.write("Il giorno corrispondente al numero" + numero\_giorno + "non esiste!");

}

else {

document.write("Il giorno della settimana corrspondente al numero" + numero\_giorno + "e' " + giorno[numero\_giorno - 1]);

}

} // Chiude la graffa aperta dopo function

</script>

</head>

<body onload = "test();">

</body>

</html>

**RETI**

**Dalla rete locale al cloud computing**

Le reti nascono private, per collegare solo determinati computer.

Nasce una necessità di un protocollo comune, per ampliare i profitti.

Rete : è un insieme di computer collegati fra loro in modo da consentire la condivisione di dati, come ad esempio file e di risorse.

risorse : stampanti, plotter…

Necessità di dispositivi per la gestione del traffico di rete (hub, switch, router, gateway)

3 caratteristiche : velocità e ampiezza di banda (banda larga 5G)

l’importante è la quantità di dati

PROTOCOLLI : un protocollo di rete è un insieme di regole utilizzate dalle due macchine per scambiarsi informazioni e specifica cosa deve essere comunicato, in che modo e quando.

Parte del software permette di comprendere tutti i linguaggi di rete.

**TIPOLOGIE DI RETE**

LAN (Local Area Network)

Connette un piccolo numero di computer 3 tipologie:

ANELLO

STELLA

BUS

Ognuno funziona da ripetitore, ma se uno si rompe l’anello si spezza e nessuno funziona.

Non è più locale, ma globale, cavi sottomarini (primo cavo sottomarino : metà ‘800).

Migliori dei satelliti.

ROUTER

se salta è un problema

Vanno bene anche per WAN (Wide Area Network) connette più LAN.

Collegamenti tra LAN : GATEWAY

**SISTEMI APERTI**

Protocolli che consentono a più dispositivi e sistemi di comunicazione.

L’ISO ha stabilito il modello OSI ( Open Systems Interconnection) per facilitare lo sviluppo di tecnologie di rete.

7 livelli di interazione:

Applicazione

Presentazione

Sessione

Trasporto

Network

Data Link

Fisico (cavi)

**Protocolli di rete :**

SMPT : posta

FTP : dropbox

TELNET : internet

**INDIRIZZI DI RETE**

Consente di comunicare, è personale in ogni dispositivo.

Stringhe separate da punti.

IPV 4 coesiste con IPV 6