Informatica
Prof. A. Frosini

11/03/20 e 12/03/20

Tecnologia informatica (Information Technology):

E' costituita da apparecchi che permettono la ripetizione/svolgimento di compiti e calcoli in tempi brevi, e che prevedono l'uso di un processore, quella parte del calcolatore che ci permette di fare computazioni (= calcolo o valutazione eseguiti con metodi e a fini determinati) che seppur limitate sono sufficienti per le nostre esigenze. La nostra attuale civiltà informatica nasce grazie all'utilizzo su larga scala del **computer**, un dispositivo che dal punto di vista logico ci permette di compiere una serie di azioni in breve tempo e al posto nostro, mentre dal punto di vista fisico è costituito da una serie di circuiti elettronici in cui è possibile misurare il passaggio di corrente (circuito aperto, 0) o l'assenza di questo passaggio (circuito chiuso, 1) chiamati anche **bit**, unità minima d'informazione all'interno di un elaboratore. Per poter far elaborare qualsiasi informazione (suoni, numeri, immagini) ad un calcolatore, occorre codificarle nel linguaggio binario (cioè ***digitalizzarle***) attraverso stringhe binarie (di 0 e 1) tramite cui l'informazione passa, viene elaborata e restituita. Il processo di codifica dell'informazione non è semplice, e può avvenire senza perdita d'informazione (es. per numeri interi) o con perdita d'informazione (es. immagini). Per poter convertire un numero dal sistema decimale (cifre in base 10) a quello binario (in base 2) occorrono alcune operazioni matematiche. Nel codice binario dobbiamo rappresentare non solo numeri, ma anche lettere, incluse quelle maiuscole, parentesi, punteggiatura e operatori aritmetici; inizialmente fu fatto usando 7 bit (n° possibili di stringhe di 0 e 1 = 2^7 =128), poi venne esteso a 8.
Il metodo di codifica più diffuso è il metodo **ASCII** (American Standard Code for Information Interchange) che permette di associare a ciascun simbolo della tastiera un codice di 7 o 8 cifre (solitamente 8, il primo dei quali è sempre 0).

Digitalizzazione delle immagini:

Considero una griglia da porre sopra l'immagine, poi prendo ogni quadratino (che nello schermo chiamo pixel) avente valore 0 o 1 in base al fatto che più della metà del suo spazio sia occupato dall'immagine (quadratino vuoto o poco occupato = 0, altrimenti 1). Ottengo una matrice (= configurazione rettangolare) di elementi 0,1 che poi posso ridurre in stringa partendo dal quadratino in basso a sx. Quella che si ottiene nella codifica è un'approssimazione della figura originaria (perché comporta una perdita di informazioni) e la rappresentazione sarà più fedele all'aumentare del numero dei pixel, poiché diminuiscono le dimensioni dei quadratini della griglia in cui è suddivisa l'immagine.
Se invece di un'immagine in bianco e nero volessi digitalizzarne una a colori avrei bisogno per ciascun pixel non più di un bit solo (0,1 = bianco e nero), ma di più bit: se voglio un'immagine con diversi livelli di grigio, ad esempio 16, ho bisogno di 4 bit (cioè di 4 stringhe binarie), poiché $2^4 = 16$. Più le immagini presentano colori più è necessario individuare un certo numero di sfumature differenti e codificare ciascuna di esse attraverso un'opportuna sequenza di bit; la rappresentazione di un'immagine mediante la codifica dei pixel viene chiamata codifica bitmap. Per capire quanto spazio occupa un'immagine (cioè quanto è grande) devo conoscere la sua risoluzione (cioè quanto è grande in numero di pixel, come numero di righe e colonne) e per ciascun pixel quanti bit vengono usati per la codifica del colore corrispondente. La grandezza dell'immagine è il prodotto tra numero di pixel x numero di bit su ciascun pixel.
Un monitor che ha risoluzione di 1280x1024 ha lo schermo diviso in 1280 righe e 1024 colonne.
Il termine risoluzione equivale a grandezza dell'immagine.

Passaggio numero decimale-binario e viceversa:
Da un numero decimale è possibile passare **al corrispondente binario** tramite una serie di divisioni successive per 2, nelle quali si considerano tutti i resti e il quoziente finale.
Da una stringa binaria è possibile ottenere il **corrispondente numero decimale** moltiplicando ogni bit della stringa per la potenza di 2 corrispondente all'indice della cifra considerata e sommando tutti i risultati.

Tipologie di computer:
-Supercalcolatori → hanno un'enorme potenza di calcolo e mantengono molti dati (es. apparecchi per le previsioni metereologiche, per la bioinformatica o per l'ingegneria aereospaziale);
-Mainframe → presentano prestazioni notevoli, elevate capacità di gestioni periferiche e buona velocità di calcolo, ma utilizzano terminali "stupidi" cioè che non effettuano alcuna elaborazione, perché non avendo una CPU non possono eseguire i calcoli (es. usati nell'ambito bancario e assicurativo, grandi aziende ed enti pubblici);
-Minicalcolatori → piccoli mainframe che usano terminali "stupidi" (es. gestione di filiali di banche, medio-piccole imprese o contabili e amministrative);
-Workstation → pc più potenti di quelli casalinghi (es. ambito ingegneristico e scientifico o dedicati al CAD);
-Personal computer → calcolatori destinati ad uso personale che hanno terminali "intelligenti" (es. ambito aziendale o familiare, server di rete, stazioni multimediali, laptop (pc portatili));
-Network computer → computer con minime capacità di memoria e limitata capacità di calcolo. Possono contare sulla potenza dei server di rete ai quali sono collegati e sulla loro possibilità di immagazzinare dati (es. smartphone, tablet).

Hardware:
E' presente in ogni calcolatore sopracitato; l'**hardware** (componente fisica) è l'insieme delle componenti fisiche del sistema.
L'architettura dell'hardware è molto complessa: la ***macchina di Von Neumann*** è un modello (che non fu progettato, solo ideato) semplificato dei calcolatori moderni; egli progettò intorno al 1950 il primo calcolatore con programmi memorizzabili anziché codificati mediante cavi e interruttori.
Presenta **5** componenti funzionali:
1) Processore (CPU) → è l'unità centrale di elaborazione delle informazioni contenute nella memoria principale, ma svolge anche funzioni di controllo. L'elaborazione avviene in accordo a sequenze di istruzioni (istruzioni macchina) e il linguaggio in cui queste vengono scritte viene chiamato linguaggio macchina; il ruolo del processore è quello di eseguire programmi in linguaggio macchina. Insieme ai dati la CPU carica anche cosa si deve fare con essi (es. somma di due numeri: preleva dalla memoria i due numeri e dice di sommarli). Un programma è una specifica univoca di una serie di operazioni che l'elaboratore deve svolgere ed è costituito da una sequenza ordinata di istruzioni macchina. Fisicamente la CPU è un microchip integrato formato da una piccola piastra di silicio sulla cui superficie sono stati creati milioni di transistor miniaturizzati, ed è costituito da varie componenti che svolgono azioni diverse (prelievo dei dati, immagazzinamento, elaborazione etc.). Queste componenti principali sono **3** (UC, ALU, Registri):
-Unità di controllo (**UC**) → esegue operazioni finalizzate al trasferimento dati o al controllo dell'esecuzione dei programmi. Si occupa di coordinare le diverse attività che vengono svolte all'interno del processore, il quale le esegue in modo ciclico, basandosi sul tempo di clock (frequenza dei processori attuali tra i 500 MHz e 3 GHz): ad ogni ciclo corrisponde l'esecuzione di un'istruzione macchina, e la UC controlla e coordina diverse attività che vengono svolte, chiamate ***fetch-decode-execute***:

-Fetch → legge, cioè carica dalla memoria principale la prossima istruzione da eseguire;
-Decode → decodifica l'istruzione e carica eventuali dati dalla memoria nei registri;
-Execute → l'ALU esegue l'istruzione;
Infine si memorizza un eventuale risultato (informazione elaborata) in memoria.
La frequenza di clock determina la velocità della CPU; l'**Hz** mi dice **quante istruzioni di una determinata cosa vengono eseguite in 1s**, è l'unità di misura della frequenza. Uno steso processore può eseguire calcoli a diverse velocità;
-**ALU** (**Unità logico aritmetica**) → è costituita da un insieme di circuiti che le permettono di eseguire operazioni matematiche (somma i bit, numeri binari) e logiche (confronto bit a bit) sui dati che sono contenuti nei registri: legge i dati contenuti nei registri generali, esegue le operazioni e memorizza il risultato in uno dei registri generali. In alcuni elaboratori oltre alla ALU si può avere un processore specializzato per effettuare operazioni matematiche particolari, il *coprocessore matematico*;
-**Registri** → unità di memoria estremamente veloci, corrispondenti a celle interne alla CPU che devono contenere le informazioni di necessità immediata per il processore e cosa esso deve farci, ovvero l'istruzione da eseguire, i dati da elaborare, i loro risultati e le informazioni accessorie (es. eventuali anomalie generate dall'esecuzione) sullo stato della CPU. Lo **stato della CPU** è la sequenza binaria determinata dalla lettura di uno o più registri all'interno della CPU. Le dimensioni di un registro sono di pochi byte (4, 8) e ne esistono di 2 tipi:
1) Registri speciali → utilizzati dalla UC per scopi particolari;
2) Registri di uso generale → registri aritmetici.
Tutte e tre queste componenti del processore dialogano con la RAM attraverso dei bus particolari: **bus dati**, **bus indirizzi** e **bus controllo**. Le operazioni della CPU sono scandite dal **tempo di clock**, una specie di metronomo estremamente veloce che dà la frequenza con cui la CPU fa una singola operazione (es. lettura/scrittura di un dato, elaborazione cioè somma/confronto di dati, che sono le uniche operazioni che può fare il processore); al secondo vengono svolte milioni di operazioni;
2) Memoria centrale (RAM) → è una sequenza di celle di memoria ciascuna contenente una parola, e ha lo scopo di memorizzare e fornire l'accesso a dati e programmi ***in esecuzione*** che stanno per essere elaborati dalla CPU. La parola è una sequenza di bit, e le parole di uno stesso calcolatore hanno la medesima lunghezza, mentre calcolatori diversi possono avere parole di lunghezza diversa. Le lunghezze tipiche sono multiple del byte (**1 byte = 8 bit**): ci sono quindi calcolatori con 8, 16, 32, 64 bit. Ciascuna cella di memoria (= parola) è identificata da un numero detto **indirizzo della cella**, che ne specifica l'esatta posizione; è grazie ad esso che ho accesso ai contenuti. La memoria centrale ha **capacità limitate** (alcuni GB), **è volatile** (= l'informazione viene persa quando il computer viene spento, la RAM si svuota) e **l'accesso alle informazioni è molto rapido** (10^{-8} s). Include almeno due tipi di informazione: la sequenza di istruzioni che devono essere eseguite dal processore e i dati su cui tali istruzioni operano. Il termine "RAM" significa "Memoria di accesso casuale", ma questo non indica che i dati siano messi a caso, ma che al processore occorre sempre lo stesso tempo per accedere a una qualsiasi, casuale, parte della memoria. Tra il processore e la RAM si trova la **memoria cache**, una memoria ad una velocità ancora superiore rispetto a quella della memoria centrale che contiene le istruzioni eseguite più recentemente. La memoria centrale, la CPU e il bus lavorano a velocità diverse, per cui la velocità complessiva del sistema è determinata dal componente più lento: per accelerare questa interazione si usa la memoria cache (perché si evita il tempo necessario per accedere alla RAM tramite il bus). Se il processore ha bisogno di leggere un dato la cerca prima nella cache che è molto più veloce, se lì non c'è la cerca all'interno della RAM.
Gerarchia delle memorie per velocità: registri, cache, RAM, dischi magnetici, nastri magnetici: più una memoria è grande più è lento l'accesso.

La **ROM** è una "Memoria di sola lettura" (in realtà principalmente lo è, ma non del tutto) situata sulla scheda madre e il cui contenuto è stato registrato in fase di costruzione del computer, per questo motivo non può essere modificata. Ogni volta che viene acceso, il computer esegue un piccolo programma contenuto nella ROM che:
-Identifica il processore;
-Controlla la quantità di RAM e ne verifica il funzionamento;
-Esamina l'hard disk ed eventuali periferiche aggiuntive;
-Legge il settore dell'hard disk in cui sono contenute le istruzioni per l'avvio del sistema. La parte adibita all'avvio del sistema si chiama **bios**, al cui interno vi è il programma di boostrap che in fase di avvio del pc oltre ad effettuare test diagnostici di base controlla lo stato delle periferiche collegate per permettere il caricamento del sistema operativo, caricando infine nella RAM, la **kernel**, ovvero la parte principale del sistema operativo.
3) Memoria di massa (o memoria secondaria) → memorizza permanentemente i dati e i programmi anche in gran quantità, non è volatile e l'accesso all'informazione non è rapido (es. hard disk, usb, floppy disk, CD-ROM, DVD-ROM). Per permettere la memorizzazione permanente deve essere costruita con tecnologie basate sul magnetismo (dischi e nastri magnetici) o sull'uso dei raggi laser (dischi ottici). Nel primo caso si sfrutta l'esistenza di sostanze che possono essere magnetizzate: la magnetizzazione può essere di due tipi (positiva e negativa) e corrisponde ai bit 0 e 1, mentre nel secondo caso si sfrutta la diversa riflessione di un raggio laser su superfici diverse aventi eventualmente anche piccoli fori. Ogni unità di superficie può essere forata o no e questo corrisponde a ai due bit 0 e 1. Il processore non può usare direttamente la memoria di massa per l'elaborazione dei dati, e il programma in esecuzione deve trovarsi nella RAM, per cui le informazioni devono essere trasferite dalla memoria secondaria alla principale ogni volta che servono.
Il disco rigido (hard disk) è caratterizzato da una serie di dischi organizzati in settori e tracce: quando necessito di un dato vado a leggere una determinata traccia/settore e gli *arm*, testine che modificano la polarità magnetica delle singole particelle per rappresentare i numeri binari, e che posizionandosi avanti e indietro vanno a leggere i dati: tutto questo avviene meccanicamente, per cui l'accesso ai dischi è molto lento. La formattazione dell'hard disk permette di cancellare tutti i dati presenti in esso tranne eventualmente il sistema operativo; possiamo poi ripristinare settori e tracce. La sua velocità di rotazione può variare da 60 giri al secondo fino a 10000 e in media un HD attuale ha velocità inclusa tra 4500 e 7200. Il tempo medio per l'accesso ai dati di un hard disk è di circa 10ms ed è dato dalla somma del:
-Tempo di posizionamento (seek time) → spostamento della testina in senso radiale fino a raggiungere la traccia desiderata;
-Tempo di latenza (latency time) → attesa che il settore desiderato si trovi a passare sotto la testina. Questo dipende dalla velocità di rotazione del disco;
-Tempo di lettura → tempo necessario alla testina per leggere/scrivere dati sul disco.
Nei **dischi ottici** quasi tutte le unità consentono solamente operazioni di lettura perché la scrittura è più complessa, in quanto richiede delle modifiche fisiche del disco (es. **CD ROM**). Nei CD ROM i bit sono codificati come aree incise (pit) e non incise (land) sulla superficie del disco; per leggere l'informazione si usa un laser di bassa potenza, i pit e i land hanno angoli di rifrazione diversi e perciò si possono distinguere; la loro capacità è di circa 650Mb. Simili ai CD ROM sono i DVD ROM che adempiono alle medesime funzioni, l'unica differenza è la maggiore capacità di memoria. Nei casi in cui le unità consentano la scrittura, questi dischi generalmente possono essere scritti una sola volta perché le modifiche fisiche che vengono prodotte sono irreversibili (es. **CD WORM**). Il raggio laser sia legge che masterizza (cioè scrive): nel primo caso lo fa grazie ai differenti angoli con cui viene riflesso dalle diverse superfici, nel secondo facendo dei piccoli buchini. Questo tipo di

laser è estremamente focalizzato e può essere emesso in fasci di dimensioni molto ridotte.
I **drives** sono i dispositivi in cui si inseriscono i floppy disk, i cd o altri dischi e contengono una testina di lettura/scrittura tramite cui avviene il trasferimento dei dati fra disco e macchina. Esistono vari modelli di drives per CD che differiscono per la velocità di lettura (es. 1X = velocità di lettura dei normali cd audio; 32X indica una velocità 32 volte superiore).
4) Dispositivi di ingresso/uscita (I/O) → componenti di collegamento con le periferiche del calcolatore (es. terminali: tastiere, mouse, monitor…) che permettono l'ingresso dei dati all'interno dell'elaboratore o la loro uscita dall'elaboratore (es. mouse, tastiera, stampante). Sono **dispositivi "stupidi"**, solitamente hanno limitata autonomia rispetto alla CPU (sono completamente gestiti, controllati e coordinati dal processore), e come le memorie di massa sono collegati a dei circuiti detti ***controller***, che gestiscono il coordinamento tra processore, memoria e dispositivi in modo da garantire il corretto trasferimento di dati. L'unità centrale (*case*) ha degli ingressi dette "*porte*" in cui inserire i cavi che collegano i dispositivi di I/O. Le porte dell'unità centrale sono varie e si distinguono in:
-Parallela → consente il transito in una sola direzione, dal computer alla periferica, attualmente non è neppure più presente (es. stampanti);
-Seriale → consente collegamenti con periferiche attive in cui ho uno scambio bidirezionale dei dati (es. mouse, modem);
-Seriale USB → permette di collegare i dispositivi anche a caldo, ovvero quando il computer è acceso;
-Serial ATA → usata per connettere hard-disk, cd, dvd a gran velocità, solitamente si trova sulla scheda madre, talvolta è esterna.
Alcuni esempi di questi dispositivi di I/O sono:
-Il **mouse** → dispositivo di puntamento che permette di trasferire un movimento su base solida lineare in uno analogo da parte di un indicatore sullo schermo del monitor detto puntatore: lo spostamento fisico del mouse viene comunicato al processore che produce lo spostamento corrispondente della freccia sul video; una volta raggiunta la posizione desiderata, premendo uno dei pulsanti del mouse si genera un segnale in input che può corrispondere a diverse funzioni.
Esistono vari tipi di mouse:
1) Mouse meccanici → la rotazione di una sfera viene trasmessa tramite sensori interni al processore. Sono molto economici ma si sporcano facilmente;
2) Mouse ottici → inizialmente usavano un LED e un trasduttore ottico-elettrico (fotodiodo) per rilevare il movimento relativo alla superficie d'appoggio, oggi incorporano un chip per l'elaborazione dell'immagine, in modo da poter essere utilizzati su un maggior numero di superfici comuni;
3) Mouse laser → sono essenzialmente mouse ottici che utilizzano un laser al posto di un LED per l'illuminazione del piano d'appoggio. Come conseguenza si ha una maggiore risoluzione nell'acquisizione dell'immagine che si traduce in migliore precisione e sensibilità di movimento.
-**Tastiera** → non ha capacità di elaborazione, è solo capace di avvertire il processore ogni volta che un carattere è disponibile in ingresso, per questo si tratta di un ***dispositivo di ingresso a carattere***. È compito del sistema prelevare il carattere, depositarlo in una memoria temporanea e, al termine dell'immissione passare i dati di input raccolti in essa al programma cui erano destinati. La tastiera è un ***dispositivo di input cieco***, in quanto l'utente non può vedere i dati immessi nel calcolatore. Per questo motivo viene usata insieme ad un dispositivo di output (schermo) su cui vengono visualizzate le informazioni. La tastiera e il video non sono direttamente collegati tra loro: è compito del processore riprodurre sul video tutte le informazioni fornite in input tramite la tastiera.
-Il **monitor** (o video) → dal punto di vista fisico può essere visto come una matrice di punti illuminati con diversa intensità. Ogni punto sullo schermo si chiama pixel e ogni immagine viene

composta accendendo o spegnendo i pixel sullo schermo. Oggi sono comuni monitor con un numero di colori che va da 256 fino a 16 milioni; ne esistono a vari livelli di risoluzione (cioè con diverse intensità di pixel), quelli attuali possono arrivare ad un'altissima risoluzione pari a 4096x3300; più il monitor è grande, e più avrà bisogno di un maggior numero di pixel per avere immagini nitide. La dimensione di un monitor viene misurata in pollici e fa riferimento alla lunghezza della diagonale. Il suo componente principale è il **display**, quel dispositivo elettronico adibito alla visualizzazione. In base alla tecnologia usata si distinguono i diversi tipi di display:
-A tubo catodico (video CRT) → il tubo illuminava i pixel dell'immagine;
-Al plasma;
-A cristalli liquidi (video LCD);
-A LED.
L'immagine che vediamo sul monitor, opportunamente codificata, viene immagazzinata in una memoria specializzata detta memoria video (VRAM), che è parte del controller della scheda grafica;
-Le **stampanti** → ne esistono di vario tipo:
1) Laser → hanno una tecnologia simile alle fotocopiatrici: stampano velocemente e silenziosamente e offrono la migliore qualità. Un raggio laser scalda e fonde sulla carta la polvere d'inchiostro e poi il foglio viene ripulito;
2) A getto d'inchiostro → sono più lente ed economiche delle precedenti ed offrono una minore qualità grafica;
3) Ad aghi → ormai quasi del tutto scomparse, usano una serie di piccoli aghi con inchiostro per formare scritte e piccoli disegni.
5) Bus → dispositivi di collegamento costituiti da piste di rame attraverso cui passano dati e informazioni di controllo tramite impulsi elettrici. I bus sono tracciati sulla scheda madre e la loro ampiezza è determinata dal numero di linee; oggi possono essere a 32 o 64 bit. Il bus che collega la CPU agli altri dispositivi del computer, fra cui la memoria centrale, è detta ***system bus***; in ogni istante il bus collega due unità funzionali: una trasmette i dati e l'altra li riceve, e questo processo viene controllato dall'unità centrale di elaborazione.

13/03/20

Software:
Software (componente logica) → insieme dei programmi che vengono eseguiti dal sistema. I programmi e i dati sono organizzati in file, ovvero archivi combinati secondo specifici criteri e residenti in memoria. I file di dati contengono informazioni (testi, numeri, immagini, suoni) mentre i file di programmi contengono sequenze di istruzioni.
Tipi di software:
-**Sistema operativo (SO)**, (Windows, Unix, Linux) → insieme di programmi che gestisce il funzionamento di base del computer (es. aspetto grafico di quello che vediamo, scrittura/lettura dei dischi, esecuzione/chiusura dei programmi) e funge da traduttore tra hardware e software applicativi es. mentre scrivo su Word ho necessità di salvare il mio lavoro: quando salvo, Word chiede a una parte del sistema operativo di salvare i dati, non è Word che interagisce direttamente con la memoria. **Il SO** risiede sull'hard disk (come tutti gli altri programmi) e viene caricato nella RAM all'accensione della macchina; **rimane sempre attivo** dal momento in cui viene caricato nella memoria centrale (all'accensione) fino allo spegnimento.
I livelli di un pc sono divisi in *livello utente* e *livello nucleo*: quest'ultimo interagisce con le varie parti del sistema operativo, è più interno.
Componenti di un sistema operativo:
1) Gestore dei processi (programmi in esecuzione) → un processo è un programma di esecuzione e pertanto necessita di determinate risorse per portare a termine il suo compito: tempo di CPU,

memoria file, dispositivi I/O. Il SO è responsabile della gestione dei processi e delle seguenti attività coinvolte in essa:
-Creazione/cancellazione dei processi;
-Sospensione/riesumazione dei processi;
-Fornire meccanismi per la sincronizzazione dei processi, la comunicazione tra loro e per evitare, risolvere e prevenire i *deadlock*, cioè tutte quelle situazioni che possono inficiare il corretto utilizzo di un processo bloccandolo o facendolo terminare.
Supponiamo di trattare sistemi monoprogrammati (detti anche mono-tasking): in essi è possibile eseguire un solo programma alla volta, i quali devono quindi essere svolti in modo sequenziale e si può mandare in esecuzione un programma solo quando il precedente ha terminato l'esecuzione.
Qualunque processo alterna fasi di esecuzioni ad altre in cui è bloccato in attesa di qualche evento esterno: può attendere che sia terminata un'operazione di input di dati oppure di poter usare una risorsa in quel momento occupata. Questi tempi devono essere ridotti al minimo, così come i tempi morti di attesa del processore. In un'esecuzione sequenziale, mentre il processo attivo è bloccato in attesa di eventi esterni, la CPU rimane inattiva in uno stato chiamato **idle**, e risulta perciò sotto-utilizzata. Per ridurre tali tempi idle, il processore realizza un parallelismo temporale svolgendo contemporaneamente i programmi: è quello che accade nei sistemi multi-tasking, dove il numero dei processi attivi viene detto *grado di multiprogrammazione del sistema*. Dal punto di vista della CPU, in ogni istante vi è un solo processo in esecuzione, se l'alternanza tra i processi è frequente, l'utente ha l'impressione che l'esecuzione dei programmi sia veramente simultanea. **Un processo può trovarsi quindi in 3 diversi stati**:
1) In esecuzione → quando sta utilizzando il processore;
2) In attesa (bloccato) → quando è in attesa del verificarsi di un evento esterno;
3) Pronto → quando è potenzialmente in condizione di poter usare il processore che è occupato da un altro processo.
Il ciclo di un processo: il processo viene creato ed è subito pronto, poi passa in esecuzione e da questo punto può essere nuovamente pronto o stare in attesa (se ha finito quello che doveva fare e può accettare nuovi input, es. ho finito di salvare i dati e posso continuare a scrivere); se è in attesa può tornare ad essere pronto quando sono terminati gli I/O e il ciclo riparte.
Il multi-tasking funziona efficacemente se il SO è in grado di offrire alcune assicurazioni:
-Il cambio di contesto → deve essere invisibile al processo;
-Quando il processore riprende l'esecuzione del programma essa deve essere ripresa esattamente nel punto in cui era stata interrotta;
-Il processore impiega un certo tempo per l'esecuzione, per cui ogni n secondi sospende l'esecuzione di un processo per passare ad un altro: questo viene chiamato ***gestione Round-Robin***, mentre la gestione sequenziale dei vari processi (= uno per volta) si chiama **FIFO** (First in first out).
2) Gestore della RAM (memoria principale) → il SO tramite il gestore della memoria principale è responsabile delle seguenti attività:
-Tenere traccia delle aree di memoria correntemente utilizzate e dei rispettivi utenti;
-Decidere quale processo caricare in memoria quando si rende disponibile dello spazio;
-Allocare o deallocare (= liberare) spazio in memoria a seconda delle richieste.
3) Gestore della memoria secondaria (hard disk, dischi, chiavette usb) → come memoria secondaria i calcolatori spesso usano i dischi e il SO li gestisce attraverso le seguenti attività:
-Gestione dello spazio libero;
-Allocazione dello spazio;
-Schedulazione dei dischi, cioè guarda quali parti della memoria del disco sono libere e va a scriverci i dati. La schedulazione (come il sequenziamento) è un processo decisionale che consiste

nell'allocare risorse finite in modo tale che un dato obiettivo venga ottimizzato.
C'è anche una gestione della memoria secondaria ***smart***, che consiste nell'allocare il più vicino possibile in maniera tale da ridurre i tempi di latenza e di movimento dell'hard disk per diminuire i tempi di accesso e caricamento della RAM;
4) Gestore delle Periferiche (dispositivi di I/O: mouse, tastiera, stampante) → il SO gestisce le operazioni sulle periferiche I/O tramite:
-Un *sistema di memoria a buffer* (**buffer** = area di memoria usata per conservare temporaneamente i dati da trasferire ad un dispositivo);
-L'interfaccia per il gestore del dispositivo di I/O (interface driver) es. proprietà che l'utente può settare per stampare fronte/retro, velocità del mouse etc.;
-I driver stessi per ciascun dispositivo di I/O (controller) → i driver sono i file di configurazione.
5) Gestore dei file e del file system → è estremamente importante; un file è una sequenza di informazioni correlate, cioè tutte appartenenti a una stessa entità logica, organizzate secondo un certo formato e memorizzate su supporti di memoria secondaria: comunemente rappresentano programmi e dati. Nella testa e nella coda dei files ci sono alcune informazioni importanti, nel caso di una foto ad esempio sono quelle inerenti a colori e grandezza. Nei files abbiamo anche un'**estensione**, 4 o 5 simboli, il primo dei quali è un punto a cui segue un'estensione (.txt, .doc, .jpg) che ci dà un'idea del contenuto e nei sistemi Windows anche quale programma dovrà gestire quel file. Il SO è responsabile di:
-Creazione/cancellazione di files;
-Creazione/cancellazione di directory (= cartelle, cioè contenitori per insiemi di file);
-Supporto di primitive (= semplici funzioni (es. search) e programmi) per la gestione di files e directory;
-Allocazione dei files nella memoria secondaria;
-Salvataggio dei dati su supporti non volatili.
Il **file system** è quella parte del SO che si occupa di gestire e strutturare le informazioni memorizzate su supporti permanenti attraverso delle cartelle che esplicitano delle relazioni di questi file. Deve fornire quindi una visione astratta dei file su disco su cui l'utente deve avere la possibilità di:
-Identificare ogni file con un nome (filename es. .txt) astraendo completamente dalla sua memorizzazione fisica (cioè dall'intero indirizzo di memoria con cui il file è stato memorizzato);
-Avere un insieme di operazioni per lavorare sui file: crearli/rimuoverli, copiarli etc.;
-Effettuare l'accesso alle informazioni mediante operazioni che non tengono conto del tipo di memorizzazione (= accedere ad un file memorizzato sul disco rigido oppure su un cd allo stesso modo);
-Avere la possibilità di organizzare i file in sottoinsiemi (sottocartelle);
-In un sistema multi-utente inoltre, l'utente deve avere dei meccanismi per proteggere i propri file.
Un insieme di operazioni minimali, presenti in tutti i sistemi è dato da:
-Creazione/cancellazione/copia/stampa di un file;
-Visualizzazione, lettura e modifica del contenuto di un file;
-Ridenominazione di un file;
-Visualizzazione delle caratteristiche di un insieme di file (dimensione, data di creazione etc.).
Il file system presenta un'organizzazione gerarchica (ad albero), poiché il numero di file che devono essere memorizzati su un disco può essere estremamente elevato e si ha quindi la necessità di mantenerli in una forma ordinata. Un unico spazio di file è scomodo perché le operazioni di ricerca e di creazione di un nuovo file diventano onerose (non è possibile avere due file con lo stesso nome). L'idea perciò, è quella di raggrupparli in sottocartelle, e tutti i SO forniscono operazioni per creare delle directory (cartelle). Una directory è costituita da un insieme di file, i

nomi dei quali sono locali alle directory (si possono avere due file con lo stesso nome purché siano in due directory diverse). In questo modo l'indice conterrà due tipi di oggetti: nomi di file e nomi di directory. Senza la strutturazione in directory, tutti i file potrebbero essere identificati semplicemente mediante il loro nome; nel caso di un'organizzazione gerarchica a più livelli si deve specificare l'intera sequenza di directory che lo contengono (per i sistemi Windows a partire dalla cartella che sta alla base di tutte le altre, ovvero la radice: C: che identifica l'hard disk, mentre E: identifica i cd e i dvd), poi i nodi sono costituiti dalle directory mentre le foglie dai file. Tra una directory e un'altra viene inserito lo slash come separatore; es. percorso che va dalla radice al file corrispondente: **C:\Biblioteca\Narrativa-Ita\libro1** → tale sequenza si chiama **pathname assoluto** del file ed è il percorso che si deve compiere per raggiungerlo a partire dalla radice dell'albero. Il **pathname relativo** invece è il percorso che si deve compiere per raggiungere il file a partire dalla directory nella quale siamo (directory corrente): se sono nella directory "Biblioteca" per raggiungere il file "libro1" → Narrativa-Ita\libro1
Per organizzare gerarchicamente i file, il SO deve fornire all'utente anche un insieme di operazioni sulle directory, tra cui quella di creare/rimuovere/rinominare una directory, elencarne il contenuto oppure spostare/copiare i file da una directory ad un'altra.
6) Gestore dei sistemi di protezione → degli accessi al sistema e alle risorse da parte di programmi, processi o altri utenti, e per garantirlo il SO deve:
-Distinguere tra uso autorizzato/non autorizzato;
-Fornire un modo per specificare i controlli da imporre → deve essere possibile mettere dei blocchi alle mie directory;
-Forzare utenti e processi a sottostare ai controlli richiesti.
La gestione dei sistemi di protezione si riferisce solo parzialmente ai virus presi navigando su Internet;
7) Gestore del networking (sistemi distribuiti) → un sistema distribuito è una collezione di processori che non condividono memoria e clock e sono connessi tramite una rete di comunicazione. Il SO deve gestire un sistema distribuito per garantire agli utenti l'accesso alle risorse del sistema, e tale gestione deve proporsi come fine l'aumento di:
-Prestazioni computazionali;
-Quantità di dati accessibili;
-Affidabilità del sistema.
8) Sistema di interpretazione dei comandi (interfaccia utente) → dà le istruzioni al SO che servono per creare/gestire processi, gestire gli I/O, le memorie, i file system etc. Il programma del SO che legge ed interpreta tali istruzioni viene chiamato *interprete della linea di comando* o *shell* o *interfaccia grafica*; la sua funzione è di ricevere un comando ed eseguirlo.
Servizi offerti dal SO:
-Esecuzione di programmi → capacità di caricarli in memoria ed eseguirli;
-Operazioni I/O → strumenti per l'uso delle periferiche → lettura e scrittura di dischi;
-Gestione del File system manipulation → capacità di leggere, scrivere, creare e cancellare files;
-Comunicazione → scambio di informazioni fra i processi eseguiti sullo stesso computer o su computer diversi;
-Determinazione degli errori → assicurare una corretta computazione determinando gli errori nella CPU, nella memoria, nelle periferiche e nei programmi.
Servizi addizionali del SO:
-Allocazione delle risorse del sistema → fra i vari utenti e i vari programmi che usano contemporaneamente il sistema;
-Contabilità → mantenimento delle informazioni relative all'uso delle risorse eseguite dagli utenti sul sistema. Gli scopi di tale operazione possono essere statistici, di protezione o di pagamento

dell'uso del sistema;
-Protezione → assicurazione che tutti gli accessi alle risorse del sistema siano controllate.
-Il **software applicativo** (Word, Excel, Photoshop) → programmi e browser che usiamo e che si appoggiano al sistema operativo (ma non fanno parte di esso) per interfacciarsi con l'elaboratore; vengono installati dall'utente per svolgere compiti specifici (elaborare dati, un testo, posta elettronica, archiviare dati etc.). Il software applicativo si divide in:
1) Utilità di sistema → programmi che servono per migliorare la gestione e la sicurezza della macchina (antivirus, compressione di file, programmi diagnostici, di backup, di ottimizzazione);
2) Strumenti di sviluppo → programmi che servono per la creazione di nuovi applicativi (programmi per la creazione di oggetti multimediali etc.)
Esistono vari tipi di software: proprietario, il cui utilizzo richiede un permesso, shareware, programmi gratuiti per un periodo limitato di tempo oltre il quale viene richiesto il pagamento, software pirata nei casi dei programmi copiati, e software freeware, quando sono gratuiti.

Scheda madre (*motherboard*):
La **scheda madre** è il supporto per la connessione di tutti i componenti interni del computer (CPU, RAM etc.) e contiene inoltre una serie di circuiti (chipset, cache, bios) adibiti al controllo delle varie parti. Ha la forma di un grande circuito stampato con dei connettori per le schede di estensione, per la RAM, la CPU etc. Vi si trovano inoltre le prese per il collegamento dell'hard disk e dei drive per i dischi mobili (cd).
Architettura della scheda madre:
E' la modalità con cui la scheda madre lavora, ovvero quella con cui scambia i dati tra CPU e periferiche inserite. In quest'ultimo caso l'architettura può essere:
-Parallela (parallelismo spaziale) → permette che in un unico calcolatore vengano eseguite diverse operazioni (processi) simultaneamente da più processori;
-Pipeline (parallelismo temporale) → permette che le operazioni vengano suddivise in stadi successivi da più componenti hardware. Se ho contemporaneamente aperto ad es. internet, word ed excel, posso mentre sto scaricando un file, andare a scrivere su word, perché in questo caso la CPU non esegue solo un compito alla volta pur avendo un solo processore, perché essa salta da un compito all'altro nei vari intervalli di tempo in cui le operazioni sono state suddivise;
-Seriale → la CPU esegue un'operazione alla volta.
Componenti della scheda madre:
1) Socket → è l'alloggio quadrato per la CPU, la quale include una serie di piedini (pin) che permettono il passaggio dati tra CPU e MB. Nelle schede embedded o in quelle vecchie e molto economiche è assente, e la CPU è saldata direttamente sullo stampato. Il socket è proprietario, cioè differisce in base alla marca (Intel, AMD etc.);
2) Slot → alloggio per la RAM e può essere di diversi tipi in base a quante sono le RAM;
3) Chipset → insieme di chip che si occupano di smistare e dirigere il traffico di informazioni passante attraverso il bus di sistema, fra CPU, RAM e controller delle periferiche di input/output;
4) ROM (Read only memory) → è la piccola memoria presente su tutti i pc che in alcuni casi può essere riprogrammata, e contiene **il BIOS** della scheda madre, un tipo di **firmware** (= software non modificabile, scritto e stampato insieme alla scheda madre e per essa). Serve per vari settaggi del pc, sia di quello dell'avvio che per quelli della ventola, a che temperatura deve sospendersi se è troppo caldo etc.;
5) CMOS → è una piccola memoria RAM in cui sono memorizzate le impostazioni del BIOS. Il CMOS è un semiconduttore che richiede pochissima energia per funzionare: quando il pc viene spento per mantenere memorizzate le impostazioni del BIOS, utilizza una piccola batteria al litio;
6) Il bus di espansione → è un collegamento dati generico progettato per permettere di collegare

alla scheda madre delle altre schede di espansione alloggiate su connettori (slot), che ne estendono le capacità (es. schede audio, video aggiuntive etc.) Le schede di espansione hanno lo scopo di espandere le funzioni della scheda madre per pilotare dispositivi interni/esterni: nel caso delle schede video su cui si connette il monitor, da esse dipendono il numero di colori del monitor, la risoluzione massima, la velocità grafica e ciò che riguarda in generale le prestazioni grafiche. Attraverso la scheda audio il computer è in grado di produrre/registrare suoni e le schede di rete riguardano le connessioni dirette alla rete (senza modem). Il **modem** (è un'altra scheda di espansione) è un dispositivo per la trasmissione e la ricezione in forma analogica o digitale: prende le informazioni digitali contenute in un computer e le converte sotto forma di suoni analogici, che possono essere inviati attraverso una linea telefonica analogica convenzionale; possono anche riconvertire in informazioni digitali i suoni analogici in arrivo. Il primo procedimento è noto come *modulazione*, il secondo come *demodulazione*: è da qui che deriva il termine "modem". La velocità con cui i modem sono in grado di scambiare i dati si misura in Kbit/secondo (Kbps), ovvero il numero di bit che il modem riesce a trasferire in un secondo. Esistono vari tipi di modem: standard (analogico), ISDN, ADSL, GSM.

Drivers:

Sono file usati dal SO per effettuare la comunicazione tra computer e le varie periferiche; per poter funzionare correttamente, ogni dispositivo deve avere il suo particolare driver registrato dal SO. Vengono installati sul computer tramite software di installazione automatica forniti assieme ai dispositivi stessi; Windows possiede una funzione (Plug and play) che all'accensione del computer verifica la presenza di nuovi componenti hardware, ricercando automaticamente il driver adatto.

Algoritmi e diagrammi di flusso:

I problemi che possiamo incontrare sono riconducibili all'elaborazione delle informazioni, per cui ciascuno è caratterizzato da un insieme di dati di partenza, da un risultato cercato e da una soluzione. La procedura di soluzione deve essere realizzata in fasi distinte e successive:
-Analisi del problema e identificazione di una soluzione da parte del primo soggetto;
-Descrizione della soluzione da parte del primo soggetto in termini comprensibili al secondo (esecutore), specificando le azioni che l'esecutore deve svolgere;
-Interpretazione della soluzione da parte dell'esecutore;
-Attuazione della soluzione da parte dell'esecutore.
Il calcolatore in quanto esecutore è caratterizzato da:
-Un linguaggio capace di interpretare con il quale devono essere descritte le soluzioni da attuare;
-L'insieme delle azioni che è in grado di compiere;
-L'insieme delle istruzioni che a ogni costrutto linguistico sintatticamente corretto associano le rispettive azioni da compiere.
Se è un problema semplice l'esecutore potrebbe svolgere la soluzione direttamente, ma in generale il problema viene scomposto in sotto-problemi finché non si raggiunge quello elementare (o primitivo), la cui soluzione corrisponde ad un'azione elementare che può essere direttamente compiuta dall'esecutore: risolvere un problema equivale quindi a risolvere una successione di problemi sempre più semplici.
L'insieme dei sotto-problemi viene risolto da una ***procedura effettiva*** quando:
-Tutti i problemi sono elementari;
-E' fissato l'ordine (cioè un flusso) di esecuzione dei problemi;
-E' specificato il modo in cui un problema utilizza i risultati di quelli che lo precedono.
Si ha **ambiguità** quando due soggetti giudicano come effettiva la stessa soluzione ma poi compiono azioni che producono risultati differenti; per rimuoverla si deve formalizzare la definizione di un

esecutore.
Gli **algoritmi** sono le soluzioni effettive, cioè che possono essere rese comprensibili all'esecutore: sono costituiti da **una successione ordinata di istruzioni** che definiscono le operazioni da compiere su dei dati **per risolvere una classe di problemi**; tali istruzioni devono rispettare i criteri di finitezza, generalità e non ambiguità.

Finitezza:
-Il numero di istruzioni deve essere finito;
-Ogni istruzione deve essere eseguita in un intervallo finito di tempo;
-Ogni istruzione è eseguita un numero finito di volte (es. non possiamo dire alla stampante di stampare all'infinito).

Generalità:
Un algoritmo deve fornire una soluzione ad una classe di problemi, prendendo dati in ingresso per elaborarli e mandarli in uscita (come una funzione): i dati in ingresso prendono il nome di "dominio" dell'algoritmo (o classe delle istanze o istanze dell'algoritmo o insieme degli input), quelli in uscita "codominio" (o insieme degli output). L'algoritmo può operare su tutti i dati appartenenti al dominio.

Non ambiguità:
Le istruzioni devono essere definite in modo univoco senza paradossi e contraddizioni, e il risultato dell'algoritmo è identico indipendentemente da chi lo sta svolgendo.
Nel caso dei calcolatori gli algoritmi sono tradotti in procedure effettive o in programmi, e il linguaggio formale per la loro descrizione è detto **linguaggio di programmazione**. Gli algoritmi sono espressi attraverso i diagrammi di flusso; le proposizioni usate dal linguaggio formale descrivono due classi principali di entità: le operazioni che devono essere eseguite e i dati.

Processo di sviluppo di un programma:
E' organizzato in varie fasi:
1) Analisi del problema e identificazione di una soluzione;
2) Formalizzazione della soluzione e definizione dell'algoritmo risolutivo;
3) Programmazione, cioè scrittura dell'algoritmo in un linguaggio di programmazione ad alto livello, cioè facile da utilizzare, in cui il mio algoritmo può essere descritto nel diagramma di flusso abbastanza naturalmente;
4) Traduzione del programma dal linguaggio di alto livello a un linguaggio macchina direttamente interpretabile dalla macchina stessa; questa traduzione può essere fatta automaticamente.

14/03/20

Algoritmi e dati:
I dati sono gli oggetti su cui si devono eseguire le operazioni e possono essere:
-**Costanti** → rimangono inalterati durante l'esecuzione dell'algoritmo da quando ha inizio a quando termina;
-**Variabili** → sono coppie < nome, valore > che possono essere immaginate come scatole aventi per etichetta il nome e per contenuto il valore. Alle variabili deve essere assegnato esplicitamente un valore (= **inizializzazione della variabile** e può avvenire da parte nostra o del calcolatore); all'inizio dell'algoritmo le variabili hanno un valore indeterminato, solo in un secondo momento prendono quello che possiamo assegnarli. Le variabili servono per riferirsi ad un valore, il quale può essere cambiato sia dal programma che da noi inserendone uno nuovo.

Assegnazione:
E' l'istruzione che permette di definire il valore attuale di una variabile, cioè quello presente in un determinato istante; rimane inalterato fino ad una nuova assegnazione. Quest'ultima viene solitamente rappresentata con: < nome ← espressione > cioè alla variabile il cui nome è "nome" io assegno il valore dato da "espressione"; espressione può essere un numero o un calcolo. Il risultato dell'espressione viene inserito (memorizzato) nelle celle di memoria etichettate con "nome".
N.B: Ogni volta che una variabile appare a destra dell'istruzione di assegnazione ←, è necessario che un valore sia stato già assegnato a quella variabile: non può partire da un valore nullo.
Es. a ← 2, b ← 3, c ← a + b allora c = 5.
Queste variabili sono scalari, poiché possono assumere un singolo valore (a prende 2, b prende 3).
Una **variabile vettore** (***array***) è una coppia < **nome, insieme di valori** > e consente di immagazzinare in una stessa variabile più elementi **omogenei** (più variabili scalari), cioè dello stesso tipo; la dimensione di un vettore è dato dal numero di elementi che contiene, le cui posizioni partono dalla 0 o da 1 in base ai linguaggi di programmazione (= gli indici vanno da 0 a N-1 oppure da 1 a N).
Es. "vettore vocali" = A E U O I → ha dimensione 5 (5 sono gli elementi, e **il nome non ha significato**, poteva chiamarsi anche "vettore vocali" e in realtà contenere numeri).
Ogni valore è individuato dal nome della variabile seguito dal numero del comparto (= dimensione) detto indice, nel caso sopracitato avremo: vocali [i] con i = numero compreso tra 0 e 4.
In fase di dichiarazione di una variabile vettore si specifica la sua dimensione che non è più modificabile successivamente in alcuni linguaggi di programmazione, in altri invece è possibile.
Assegnazione di un valore a un vettore:
L'istruzione di assegnazione a un vettore è analoga a quella per una variabile ordinaria, con la differenza che si deve specificare anche a quale posizione ci stiamo riferendo.
Es. vettore vocali di dimensione 5 possiamo scrivere: *vocali* [3] ← E (cioè "vocali 3 prende E", la lettera E viene assegnata in posizione 3 nel vettore vocali) oppure *vocali* [0] ← I (vocali 0 prende I, nella posizione 0 andremo a sovrascrivere la I).
N.B: non possiamo dare come istruzione:
-*vocali* [-5] ← U perché non si possono contare posizione negative, e neppure scrivere
-*vocali* [122] ← A perché non si possono indicare posizioni successive alla dimensione del vettore, 4 (o 5), nel nostro caso.

Matrice (M [i, j]):
E' l'estensione del concetto di vettore, in quanto insieme di valori che sono indicizzati facendo ricorso a due o più indici. Per una matrice a 2 dimensioni l'indice ***i*** è detto ***riga*** e quello ***j colonna***; l'assegnazione avviene come per il vettore, con la differenza di dover specificare gli indici. Dopo aver assegnato un indice i e uno j, andremo a trovare l'intersezione tra riga e colonna che ci darà l'elemento corrispondente. Una matrice con 3 righe e 4 colonne è una matrice 3x4; righe e colonne partono da 0 come posizione in alcuni linguaggi di programmazione (in Matlab partiranno da 1).
Es. *matr_vocali* [1, 2] ← O vuol dire che la O sarà sovrascritta nell'intersezione di riga 1 e coloa 2.

Istruzioni:
Le istruzioni che possiamo usare per manipolare i nostri dati si possono suddividere in **6 categorie**:
1) Istruzioni operative → producono un risultato; ne fanno parte le ***operazioni aritmetiche*** e le ***assegnazioni*** di valori a variabili (5 + 3; x ← 2; 3 mod 2 dove mod sta per "modulo", ovvero resto della divisione intera 3:2, cioè 1. L'operatore modulo può essere usato anche per riconoscere se un numero è pari o dispari e per indicare i multipli di un certo numero). Posso anche verificare se due stringhe di variabili che ho sono uguali, oppure tagliarne una parte o altre operazioni simili;

2) Istruzioni di salto → alterano il normale ordine di esecuzione delle istruzioni di un algoritmo, specificando esplicitamente quale sia la successiva istruzione da eseguire (facendo fare quindi un "salto"). Si distinguono in ***istruzioni di salto condizionato*** e ***incondizionato*** a seconda che debba essere verificata una condizione per poter attuare il salto oppure no. Nei linguaggi di programmazione "strutturati" non sono usate, non erano molto comode perché non permettevano di prevedere il flusso dell'elaborazione, per questo poi sono state eliminate;
3) Istruzioni di inizio/fine esecuzione;
4) Istruzioni di ingresso/uscita → indicano una trasmissione di dati fra l'algoritmo e tutto ciò che è esterno ad esso (es. mandare in stampa una determinata scritta, far suonare dalle casse una certa melodia). Si chiamano di ingresso o lettura quando l'algoritmo riceve dati dall'esterno, si dicono di uscita o scrittura quando i dati sono comunicati dall'algoritmo all'esterno.
5) Istruzioni condizionali → controllano il verificarsi di condizioni specificate e che in base al risultato determinano quale istruzione eseguire. **N.B**: si altera il flusso del programma in funzione della condizione (se è vera o falsa) e si presentano nella forma: se…allora…altrimenti;
6) Istruzioni di loop o di ciclo → determinano il ripetersi di una stessa serie di istruzioni fintanto che una condizione rimane verificata o non verificata: ad esempio stampo finché non raggiungo il numero di copie stabilite. Tale condizione può essere testata sia prima che dopo l'esecuzione, per la prima volta, della serie di istruzioni. Si presentano nelle forme: fintanto che…esegui… (for…do…) o esegui…fintanto che… (repeat…until… o con un while).

I predicati nelle istruzioni di controllo:
Le istruzioni di controllo devono determinare (calcolare) la verità/falsità di un enunciato, che viene chiamato proposizione; si parla pertanto, di valore di verità di tale proposizione, la quale è vera o falsa in qualsiasi istante la testi.
Es. "3 è un numero pari" è proposizione, "incrementa x di 10" no.
E' un **predicato** una proposizione che contiene delle variabili e in cui il valore di esse determina il valore di verità del predicato. La composizione di un predicato avviene attraverso operatori relazionali (>, <, =, ≠, ≤, ≥); es. la variabile età è minore di 30. Quando contiene un solo operatore relazionale è detto predicato semplice, altrimenti è detto composto. In quest'ultimo caso, significa che ho unito gli operatori relazionali con quelli logici:
-**Not** (negazione) → dato un predicato (p), NOT p è vero quando p è falso e viceversa. Il NOT è un operatore unario, perché prende un predicato semplice e ne cambia il valore di verità;
-**And** (congiunzione; &) → dati due predicati (p e q), p AND q è vero solo quando entrambi sono veri e falso in tutti gli altri casi;
-**Or** → dati due predicati (p o q), p OR q è falso solo quando entrambi sono falsi ed è vero in tutti gli altri casi.
Gli operatori or e and sono binari.

15/03/20

Diagramma a blocchi (diagramma di flusso):
E' un linguaggio formale di tipo grafico per rappresentare gli algoritmi con cui si può indicare l'ordine di esecuzione delle istruzioni; i simboli grafici che concorrono alla definizione di un diagramma a blocchi si chiamano ***blocchi elementari*** (sono associati ad ogni tipo di istruzione elementare), e sono tra loro collegati tramite frecce che indicano il susseguirsi delle istruzioni.
I blocchi elementari sono:
-**Begin** → blocco di inizio;
-**End** → blocco di fine;
-**Leggi x** → blocco di lettura (blocco di in);
-**Scrivi x** → blocco di scrittura (blocco di out);

-A → blocco di azione dove la A rappresenta una delle due istruzioni operative: o l'assegnazione di un valore a una variabile o compiere una computazione;
-C → blocco di controllo; è un blocco di selezione che mi permette di formare il costrutto if…then…else… poiché C è una proposizione o un predicato semplice/composto e in base al suo valore di verità seguiamo le frecce.
Mancano le istruzioni di salto che sono assenti in una programmazione ben strutturata e i loop: per fare i loop combiniamo i blocchi che abbiamo.
Un diagramma a blocchi descrive un algoritmo se:
1) Ha un blocco iniziale e uno finale;
2) E' costituito da un numero finito di blocchi azione e/o blocchi lettura/scrittura e/o blocchi di controllo;
3) Ciascun blocco elementare soddisfa le seguenti condizioni di validità:
-Ciascun blocco azione, lettura/scrittura ha una sola freccia entrante e una sola freccia uscente;
-Ciascun blocco di controllo ha una sola freccia entrante e due uscenti;
-Ciascuna freccia entra in un blocco o si innesta su una altra freccia;
-Ciascun blocco è raggiungibile dal blocco iniziale;
-Il blocco finale è raggiungibile da qualsiasi altro blocco, altrimenti avrei un loop.

Analisi strutturata:
Analisi volta alla stesura di descrizioni di algoritmi tramite diagrammi a blocchi di tipo strutturato, i quali sono più comprensibili e modificabili e in cui non apparirà mai un'istruzione di salto incondizionato.
Teorema di Bohm-Jacopini: ogni diagramma a blocchi non strutturato (cioè dove non valgono quelle cinque regole) è sempre trasformabile in un diagramma a blocchi strutturato ad esso equivalente. Due diagrammi sono equivalenti se partendo dagli stessi dati iniziali producono gli stessi risultati, cioè se sono equivalenti gli algoritmi (e fanno quindi la stessa cosa).
Una descrizione è di tipo strutturato se i blocchi sono collegati tramite i seguenti schemi di flusso strutturato:
1) Schema di sequenza → due o più schemi di flusso sono eseguiti in successione. Lo schema di sequenza è strutturato se e solo se lo sono i blocchi S_1 e S_2 messi in sequenza;
2) Schema di selezione → esiste un blocco di controllo che permette di scegliere quale schema di flusso debba essere eseguito tra due, in funzione del valore di verità di controllo.
3) Schema di iterazione (o di ciclo o di loop) → modo conciso per descrivere azioni che devono essere ripetute. Le condizioni vero/falso per il controllo possono essere invertite: si parla di ***iterazione per vero*** quando S_1 è eseguito finché la condizione su C è vera, e ***iterazione per falso*** nell'altro caso.
Quando è necessario eseguire lo stesso insieme di operazioni per un certo numero di volte si adotta un particolare schema di iterazione, che inizia con una sequenza di azioni di assegnazione dette *istruzioni di inizializzazione*, e possiede una sequenza di azioni che viene ripetuta per un numero specificato di volte, in quale viene indicato dalla **variabile di ciclo**.
Condizione di fine ciclo:
Viene controllata dopo l'esecuzione di ogni blocco di iterazione (sequenze di azioni), può essere con controllo **in coda** al ciclo o **in testa**. Un ciclo è detto **enumerativo** quando è noto a priori il numero di volte che deve essere eseguito. Si usa un **contatore di ciclo** (è una variabile) per controllarne l'esecuzione: si usa cioè una variabile che viene incrementata (o decrementata) fino a raggiungere un valore prefissato. Un ciclo è **indefinito** quando non è noto a priori il numero di volte che deve essere eseguito; accade quando la condizione di fine ciclo dipende dal valore di una o più variabili che o dipendono dall'interazione con l'esterno, o vengono modificate all'interno.