1. Architettura del calcolatore

L'architettura è un'organizzazione dello spazio in maniera funzionale (nell'informatica). L'hardware ci impone dei vincoli (lavorare in binario) e quindi ci impongono di lavorare in un certo modo. Il personal computer (PC), rispetto ad altri dispositivi elettronici, ha la possibilità di fare molte più cose differenti, posso sfruttarlo per scaricare programmi per il quale non è stato pensato originariamente. Un PC non ha dunque uno scopo preciso ma un "general purpose" (il PC è diverso da una lavatrice, da un bancomat ecc...). La differenza tra un telefono e uno smartphone è il fatto che nel secondo posso INSTALLARE applicazioni (non è un sistema chiuso come nel telefono). Lo smartphone è dunque "general purpose". Le componenti principali sono: Hardware e Software. Il funzionamento di un PC si basa su: istruzioni da eseguire e dati da elaborare.
Hardware: componenti fisiche e principali del pc (e dello smartphone). L'hardware senza software non funziona
Software: divisibile in software di sistema (o d'ambiente) che gestisce il calcolatore e software applicativo che è dedicato alla realizzazione di specifiche esigenze applicative. Il software applicativo è specifico per il sistema per il quale è stato progettato.

Modello di Von Neumann

Von Neumann ha studiato un modello di elaboratore quando ancora non esistevano. Il suo modello è molto semplice composto da solo quattro elementi funzionali:
- unità di elaborazione (CPU): interpreta ed esegue dei programmi e coordina la macchina
- memoria centrale: contiene dati e istruzioni
- interfacce delle periferiche: scambio di informazioni con mondo esterno (tastiere, mouse, stampanti ecc...)
- bus di sistema: collega gli elementi funzionali

Funzionamento del modello di Von Neumann: nella memoria centrale sono presenti le istruzioni e i dati
1) il processore estrae le istruzioni e le esegue (una per volta), le istruzioni possono essere operazioni di manipolazione o trasferimento di dati
2) i dati vengono trasferiti attraverso il bus di sistema
3) le fasi di elaborazione si susseguono in modo sincrono rispetto ad un orologio di sistema (clock), per es. 1.7 gigahertz è la velocità del clock
4) durante ogni intervallo di tempo l'unita di controllo (parte del processore) stabilisce la funzione da svolgere
5)l'intera macchina lavora in modo sequenziale

Memoria centrale: divisa in celle che contengono dati e istruzioni, con spazi (chiamati parole) adibiti a dati e spazi adibiti a istruzioni. Ogni parola può essere a 16 o più bit. La memoria centrale è una memoria volatile, cioè che non è permanente (se il pc si spegne senza aver salvato il contenuto va perso). Una memoria volatile è molto più veloce di una non volatile (es. hard disk). Abbiamo 1024 celle (cioè posso avere 1024 parole). La memoria centrale è dunque formata da un insieme di celle, ognuna con un indirizzo, mentre il registro dati contiene il dato da leggere/scrivere. Questa è la rappresentazione astratta della memoria RAM e ROM (la ROM è solo leggibile)
Unità di elaborazione(CPU): Formato da molti componenti: L'unita di controllo che è un modulo che preleva e decodifica le istruzioni e le esegue, il clock, ALU che esegue le operazioni aritmetico-logiche (somma, sottrazione, comparazione, ecc...), Registro dati (parola da leggere/scrivere) e Registro indirizzi (indirizzo della cella della memoria centrale), registro di stato (mi permette di stabilire se le operazioni sono andate a buon fine).
Bus di sistema: collega il registro dati e il registro indirizzi tra la memoria centrale e la CPU.
Interfacce delle periferiche: all'interno sono presenti dei registri, uno dedicato ai dati da leggere e scrivere, uno dedicato allo stato della periferica e uno dedicato al comando da eseguire

Architettura del calcolatore:

Avremo sicuramente una CPU, avremo una memoria centrale con RAM e ROM, abbiamo le interfacce delle periferiche e le periferiche, avremo il bus. L'architettura del calcolatore è la trasformazione reale del concetto della macchina di Von Neumann. Anche l'hard disk e le schede grafiche sono delle periferiche anche se sono all'interno del Case (o cabinet). Il cabinet è tutta l'infrastruttura esterna.
Scheda Madre: è la scheda centrale che fornisce tutte le connessioni con gli altri componenti (è la versione "reale" del bus del sistema). È un grande circuito stampato sul quale si trovano i componenti principali. È presente anche una batteria che serve solamente ad alimentare dei componenti che calcolano il passare del tempo. Il 1° gennaio 1970 è la data di "creazione" standard dei computer. Se la piccola batteria si scarica verrà mostrata questa data.
Microprocessore: la parte superiore è liscia, la parte inferiore presenta molti piccoli "dentini" per essere alloggiati sulla scheda madre. Al processore va sempre agganciata una ventola. La CPU si occupa dell'esecuzione dei programmi e per svolgere questo compito legge e scrive i dati sulla memoria ram.

Memoria (RAM-ROM): la memoria RAM è l'acronimo di "Random Access Memory" e mantiene le informazioni fintanto che il PC è acceso ed è il posto dove sono conservate tutte le informazioni dei programmi su cui sto lavorando. La memoria Cache è una memoria talvolta alloggiata all'interno del microprocessore molto veloce e temporanea. La Cache è molto più veloce della RAM ma anche più costosa quindi è molto ridotta di dimensioni. Le ROM sono memorie che possono solo essere lette.
Interfacce: sono le porte di comunicazione tra il computer e le periferiche. La tecnologia più diffusa si chiama USB
Hard Disk: ci permette di memorizzare permanentemente le informazioni. Aprendolo vedrei una serie di dischi rigidi impilati attorno ad un perno e ci sono delle testine che vanno ad avvicinarsi alle superfici dei dischi. Questi dischi sono ricoperti di un materiale ferromagnetico che permette la memorizzazione permanente. Questa superfice può essere polarizzata e quindi posso memorizzare due stati (un verso o l'altro). Siccome la polarizzazione è permanente anche se spengo il PC questo materiale mantiene la sua polarizzazione. Ho delle testine per ogni faccia del disco che leggono il disco mentre è in rotazione. Ogni faccia è divisa in tracce in settori concentrici, le tracce a loro volta sono suddivise in settori. Formattare vuol dire impostare questa struttura all'hard disk. Io posso accedere una specifica informazione specificando traccia e settore. Uno svantaggio è che un disco in rotazione è lento perché spostare la testina impiega tempo. Il secondo svantaggio è che mantenere dischi in perenne rotazione consuma energia.
Solid State Disk (SSD): sono dispositivi di archiviazione permanenti. L'organizzazione fisica è la stessa di un hard disk, ma non ha elementi in rotazione. Non ho del materiale ferromagnetico. Un SSD è organizzato in Plane, Block (contenitore delle pagine) e Page (contengono le informazioni). Il tempo di accesso di un SSD è molto più veloce ma la cancellazione delle informazioni su un SSD avviene a livello di blocco quindi nella pratica sono simili ad un hard disk. Inoltre, nei dischi fissi il "danneggiamento è costante" mentre gli SSD all'inizio sono molto robusti ma poi improvvisamente ho degli enormi danni. Quindi non possiamo dire che uno sia meglio dell'altro.
CD e DVD: CD audio e CD dati sono gli stessi e la quantità di dati che può essere memorizzato è 650 MB di dati. I DVD hanno la stessa logica, stessa grandezza ma possono avere un maggior numero di celle e quindi possono avere più memoria. La lettura è ottica e avviene attraverso una parte di materiale trasparente che è essenziale perché altrimenti la luce del raggio laser ha una rifrazione del raggio e la lettura sarà sbagliata.
Monitor: Un monitor viene organizzato attraverso una matrice di punti chiamati pixel. Ogni pixel è la quantità minima di informazioni visibile sullo schermo. Più è elevata la risoluzione più i pixel saranno piccoli (a parità di dimensione). Un singolo pixel è composto da tonalità di tre colori differenti. La frequenza di Refresh è la velocità con la quale lo schermo cambia (si aggiorna).

2.RAPPRESENTAZIONE BINARIA DELL'INFORMAZIONE

Le informazioni possono essere:
- immagini
- video
- caratteri e testi
- suoni
- numeri

E sono tutte rappresentate in binario, come faccio allora a gestire informazioni del mondo reale nel mondo binario?
Numeri: per scrivere programmi ho bisogno di un algoritmo. Per codificare (tradurre in un formato digitale le informazioni) ho bisogno di tre passi:
1) definisco un alfabeto dei simboli: per esempio 0,1,2,... ho separatori decimali (,), cifre(1), separatore delle migliaia(.), segno positivo e negativo (+ -)
2) regole di composizione (sintassi): definisce le sequenze di cifre (parole) ammissibili. Per esempio, 1.234,5 è la rappresentazione di un numero reale (è una parola ammissibile) mentre 1,23,45 non lo è.
3) codice (semantica): insieme di regole che per le parole ammissibili associa un'entità di informazione. 1.234,5 = 1x10^3 + 2x10^2 + 3x10^1 + 4x10^0 + 5x10^(-1). 1,23,45=?
Lo stesso alfabeto può essere utilizzato con codici diversi (in Italia la virgola ha un valore diverso rispetto all'Inghilterra)
La codifica del calcolatore è una codifica binaria, il bit è la quantità più piccola di informazione che può essere tra due stati: on/off, acceso/spento, si/no
Il calcolatore tratta diversi tipi di dati tutti rappresentati con la codifica binaria
1bit= 2 stati(0,1)= due oggetti
2bit= 4 stati (00, 01, 10, 11)= quattro oggetti
3bit= 8 stati (000, 001, 010, 100, 101, 110, 111)= otto oggetti
Kbit= 2^k stati= 2^k oggetti
Posso andare a capire quanti bit mi servono per codificare N oggetti= N<2^k à k=[log(2)N]

Byte:
il bit rappresenta 2 stati (0 o 1), con 8 bit ho 256 stati e vengono definito come byte (256 configurazioni) poi ho il kilobyte (1024 byte), il megabyte, gigabyte e terabyte
Codifica binaria dei caratteri:
parto da un insieme di oggetti che voglio configurare e ho un sistema da configurare. Per codificare i caratteri ho diversi elementi: 26 lettere maiuscole, 26 minuscole, 10 cifre, 30 simboli di interpunzione, 30 caratteri di controllo (EOF, CR,...). In totale ho circa 120 oggetti, quindi ho bisogno 7bit per codificare questi oggetti. Per questo è stato inventato il codice ASCII che utilizza 7 bit che rappresenta i caratteri (e li trasforma in codice binario). Il codice ASCII è una tabella che associa biunivocamente un carattere con un codice. Vie anche una codifica ASCII estesa che utilizza 8 bit (1byte) utilizzata dai vecchi telefoni. Vi è poi la codifica UNICODE con 16 bit (2 byte) utilizzata al giorno d'oggi.
Codifica delle stringhe:
una stringa è una sequenza di caratteri, ogni carattere è codificato uno per volta.
Numeri Naturali:
dobbiamo rappresentare un numero attraverso una rappresentazione binaria e per fare questo utilizziamo la Notazione posizionale che permette di rappresentare un numero intero naturale positivo nel modo seguente:
le sequenza di cifre c rappresentata in base B>2. La notazione convenzionale è in base 10

es.
12 (10)
B=10
Cifre 0, ..., 9

2 unità 2x1
1 decine 1x10

10 + 2 = 12
1 x B^(2-1) + 2 x B^0 = 1 x 10^1 + 2 x 10^0 = 12

Codifica Binaria:
Usata dal calcolatore per tutte le informazioni dove la base B = due, A = (0, 1)
Es.
1101 (2)
1 x 2^3 + 1 x 2^2 + 0 x 2^1 + 1 x 2^0
Conversione da decimale a binario:
utilizzo la divisione per 2 continuando fino a che il quoziente non va a 0 e guardo la sequenza di resti
es.
23 (10)
23:2=11 r=1
11:2=5 r=1
5:2=2 r=1
2:2=1 r=0
1:2=0 r=1 (si legge dal basso)

23(10)=10111 (2)
Codifica binaria su n bit:
con n bit posso rappresentare 2^n configurazioni da 0 a 2^(n-1)
Numeri binari Naturali:
con un byte:
00000000 (bin) = 0 (dec)
00001000 (bin) = 8 (dec)
11111111 (bin) = 255 (dec)
Numeri interi:
in una codifica modulo e segno , il segno più è codificato con un bit, mentre il resto dei bit codifica il numero
es.
000000000 (m e s) = + 0
100001000 (m e s) = - 8
000001000 (m e s) = + 8

In questo caso ho le configurazioni da -255 a +255 ma ho una inconsistenza nel numero 0 (che presenta 2 codifiche) per questo non viene utilizzata dai computer.

Per capire la semantica che sto utilizzando devo per forza avere il "pedice" al numero

Dentro al calcolatore:
il calcolatore moderno contiene 32 o 64 bit

RAPPRESENTAZIONE BINARIA DELL'INFORMAZIONE (parte2):
Un suono è una variazione della pressione dell'aria e questa variazione viene interpretata dal nostro cervello come un suono. Nel mondo digitale il suono è una sequenza di bit. Qualsiasi segnale audio può essere visto come la somma pesata di suoni semplici (sinusoidi).

Codifica audio:
le onde sonore sono dunque segnali continui e io devo codificarlo in un istante nel tempo finito e con un range di ampiezze finito. Per fare ciò utilizzo due tecniche:
- discretizzazione del tempo (decido gli istanti di tempo in cui fare la misura) e viene detto campionamento
- discretizzazione delle ampiezze (associo l'ampiezza in un sistema finito) e viene detto quantizzazione

Campionamento:
si fa in modo regolare dopo aver definito la frequenza di campionamento, sapendo la frequenza so anche il periodo di tempo. La scelta della frequenza è un elemento chiave. T è detto periodo di campionamento (in secondi), F=1/T è detta frequenza di campionamento (in Hz). Se ho una frequenza troppo bassa il suono risulterà sgranato ma se prendo una frequenza troppo alta occuperà una memoria molto alta.
Per esempio, per i segnali di tipo vocale (telefono) la frequenza di campionamento è di 8kHz
Per i segnali audio musicali la frequenza di campionamento è di 44.1kHz
Con un campionamento più fitto avrò una fedeltà maggiore ma dovrò usare più memoria.

Quantizzazione:
I campioni estratti con la quantizzazione rappresentano le ampiezze con precisione arbitraria (anche infinita), per poter essere rappresentato da un calcolatore, il valore dell'ampiezza deve essere espresso tramite un numero finito di bit. L'ampiezza può prendere un numero predefinito di valori, se ho 3 bit posso avere 8 configurazioni e l'ampiezza che entra come un valore continuo esce con un valore preciso e predefinito.

ESEMPIO CODIFICA AUDIO:
canzone salvata su cd
frequenza di campionamento 44100Hz
16 bit a campione
2 canali (destro e sinistro)

2 x 16 x 44100 = 176 Kbyte/sec

1 canzone di 5min:
176Kbyte x 60sec x 5min = circa 52Mbyte

Nel tempo sono nati dei formati come MP3 che pesa 5Mbye al posto di 50Mbyte attraverso un fattore di compressione di circa 10:1. Per poter comprimere una canzone perderò comunque un po' di fedeltà ma esistono comunque delle tecniche di compressione (lossless o lossy) che riducono l'occupazione di memoria. Le tecniche lossy sfruttando le debolezze dell'orecchio umano (un esempio di compressione lossy è JPG). Le tecniche lossless non perdono informazioni (es file ZIP) ma comprimono molto poco.

Codifica delle Immagini:
Le immagini, come per i suoni devono essere prima discretizzate e poi codificate sotto forma di numeri. Discretizzare vuol dire trasformare l'immagine in parti distinte e poi codifico ogni singola parte in forma numerica. Esistono 2 grosse famiglie di immagini: Scalari (o raster) come le foto e le immagini Vettoriali come un disegno geometrico.

Immagini raster:
Per discretizzare un immagine e codificarla devo sovrapporre una griglia all'immagine e identifico i pixel influenzati dalla presenza dell'immagine. Salvando l'immagine avrò solamente i pixel e non più l'immagine originale. I pixel vanno a discretizzare l'immagine. Il pixel è la più piccola unità informativa dell'immagine (sotto il pixel non posso scendere). La griglia si chiama risoluzione e proprio come uno schermo la risoluzione dell'immagine è la dimensione della griglia utilizzata per discretizzare l'immagine. A parità di dimensione aumentando la risoluzione (il numero di

pixel) diminuisco la grandezza dei pixel e l'immagine risulta migliore. Anche in questo caso (come per il campionamento della musica) con una maggiore risoluzione grafica aumenterà l'utilizzo di memoria.
Dopo aver discretizzato l'immagine devo quantizzare i colori dell'immagine. Per fare ciò bisogna rappresentare ogni pixel con un numero. Il numero rappresenterà il colore di ogni pixel. Se io ho un'immagine in bianco e nero ho solo due tipologie di colori, quindi si rappresenta con solo 1 bit: 0 per il colore bianco e 1 per il colore nero. Questo procedimento vale anche per le immagini in scala di grigio (ovvero fisso un insieme delle tonalità di grigio, di solito 256) dove ogni pixel è codificato con un byte (8 bit). Per quanto riguarda le immagini a colore utilizzo la rappresentazione dei colori secondo un modello (RGB) che si basa sulla sintesi additiva dei tre colori primari (un azzurro può essere rappresentato con la combinazione dei tre colori). Dobbiamo avere una specifica codifica per ogni tonalità con la stessa modalità utilizzata per la scala di grigio: divido il rosso in 256 tonalità, il verde in 256 tonalità e il blu in 256 tonalità. Perciò codifico un immagine a colori codificando separatamente ogni canale. Per ogni pixel ci sono 24 byte (8 bit per ogni tonalità di colore). Risoluzione e quantizzazione vanno a braccetto (non ha senso avere una risoluzione enorme per un immagine in bianco e nero).

Esempio: ho un immagine di 1024 x 768 pixel con 256 toni di grigio/pixel = 1024 x 768 pixel x 8 bit = 768 Kbyte

Compressione delle immagini:

In fase di codifica delle immagini devo comprimere queste immagini per diminuire il peso e aumentare la velocità di trasferimento. Anche in questo caso ho due tipi di tecniche di compressione: lossless (.zip), alcune pensate ad hoc per le immagini (.png) e le tecniche lossy (jpeg o gif) che, sfruttando le debolezze dell'occhio umani di essere poco sensibile a lievi cambiamenti di colore in punti contigui aumentano la compressione ma perdono anche informazioni. Generalmente attraverso alcuni parametri posso decidere quanto comprimere.

Principali formati di compressione:

TIFF: formato che utilizza 24 bit per pixel ed è una compressione senza perdita di informazioni (di solito esce dalle fotocamere professionali)
GIF: compressione senza perdita, più immagini nello stesso file
PNG: compressione lossless, studiato per sostituire il formato GIF che è protetto da brevetti
BMP: utilizzato nei programmi Microsoft (paint) senza perdita di informazioni
JFIF: compressione JPEG

Esempio: una fotocamera a 8Mpixel sono 8.000.000 di pixel se non la comprimo a colori occupa 8.000.000 x 3 componenti/pixel x 1byte/componente = 24.000.000 byte = 24Mbyte
Se salvo l'immagine in formato JPEG ottengo un file di dimensione 2.4Mbyte, ovvero il rapporto di compressione è di 10:1

Immagini Vettoriali:

A differenza del formato raster queste immagini nascono nel mondo virtuale. Queste immagini sono composte da oggetti definiti mediante relazioni matematiche tra punti e linee. Non esiste una perdita di dettaglio andando a ingrandire l'immagine perché non esiste il concetto di risoluzione, non ci sono i pixel nelle immagini vettoriali. Le immagini vettoriali sono dunque composte da oggetti e questi oggetti sono composte da una serie di primitive come punti, linee, rettangoli, ellissi, ecc…
Questo ha diversi vantaggi: sono composte da oggetti geometrici, può essere scalata a qualsiasi dimensione senza perdere qualità, non dipende dalla risoluzione, occupano poca memoria.
Ha però anche alcuni svantaggi: sono generati all'interno della macchina e perciò non è il miglior formato per immagini di tipo fotografico con toni continui, luci, ombre, miscele di colore.
È possibile passare da un immagine vettoriale a una raster attraverso la rasterizzazione, ma non è possibile l'opposto.

AutoCAD:

l'immagine prodotta è di tipo vettoriale, ci sono due formati di salvataggio: DWG, nativo di AutoCAD, e DXF, formato di interscambio tra varie applicazioni.

Formati vettoriali/bitmap (misti):

PDF: posso zoomare i file vettoriali all'interno del PDF ma posso anche inserire parti raster. Dunque, le immagini vettoriali resteranno tali così come le immagini raster.

Immagini in movimento (Video):

Il video è una successione di immagini fisse (o frame) trasmesse con velocità sufficientemente elevata. Questa velocità è di solito di 24/30/60 frame al secondo. Con un numero abbastanza alto di fotogrammi fissi al secondo l'occhio umano percepisce il movimento. Esistono due tipi di segnali video: Segnale interlacciato (frame suddiviso in righe) invio la metà di informazioni che mando raddoppiano i semi-frame. Nel segnale progressivo ogni frame è costituito da tutte le righe.

Codifica Video:

potrei codificare separatamente ogni fotogramma come un immagine fissa, questa è la tecnica utilizzata da molte fotocamere compatte che salvano i filmati ripresi in formato Motion-JPEG, ma in questo caso i rapporti di compressione sono dell'ordine di 10 (per memorizzare un film su un DVD ho bisogno di comprimere di un ordine di 50). Per ottenere rapporti di compressione più alti devo utilizzare la ridondanza temporale, ovvero per due frame consecutivi molto simili codifico solo le differenze. Con questa tecnica posso ottenere dei rapporti di compressione dell'ordine fino a 100.

ALGORITMI:

L'algoritmo è una sequenza di passi che permettono di risolvere un problema, ovvero il processo che dato un problema ed individuato un opportuno metodo risolutivo trasforma i due dati iniziali nei corrispondenti risultati finali. Questo processo può essere fatto da un umano ma a noi interessa che sia il calcolatore a risolvere questi problemi. Qualsiasi algoritmo deve essere tradotto a funzioni elementari più semplici. L'algoritmo è una sequenza finita di azioni che risolve in un tempo finito una classe di problemi. L'esecutore è una macchina astratta capace di eseguire le azioni specificate dall'algoritmo. Una ricetta è un algoritmo.

Proprietà degli algoritmi:

Eseguibilità: ogni azione deve essere eseguibile dall'esecutore in un tempo finito
Non-ambiguità: ogni azione deve essere univocamente interpretabile dall'esecutore
Finitezza: il numero totale delle azioni da eseguire, per ogni insieme di dati, deve essere finito

Elementi degli Algoritmi:

- Oggetti: le entità su cui opera l'algoritmo. Sono i dati iniziali del problema, le informazioni ausiliarie, risultati parziali e finali. Le informazioni sono dette anche dati e possono essere variabili e costanti.
- Operazioni: interventi da effettuare sui dati. Calcoli, confronti, ricopiature, acquisizioni, emissioni, ecc…
- Flusso di controllo: l'indicazione delle possibili successioni dei passi dell'algoritmo. Indica tutti i possibili percorsi all'interno dell'algoritmo. La correttezza dei risultati dipenda anche dalla sequenza con cui sono eseguite le operazioni. È la descrizione a priori di tutte le possibili sequenze nell'esecuzione dei passi dell'algoritmo.
- Flusso di esecuzione: è una particolare sequenza che dipende dai dati in ingresso. Viene eseguita durante una particolare esecuzione dell'algoritmo.

RAPPRESENTAZIONE DI UN ALGORITMO DESTINATO ALL'ESECUZIONE AUTOMATICA:

Per rappresentare un algoritmo devo avere una descrizione univoca per l'esecutore di tutte le possibili sequenze di operazioni da eseguire per risolvere il problema dato. Per poter fare questo devo poter descrivere il flusso di controllo, le operazioni eseguibili e gli oggetti si cui agiscono le singole operazioni. È necessario un linguaggio costituito da:

- Vocabolario: insieme di elementi per la descrizione di oggetti, operazioni e flusso di controllo
- Sintassi: insieme di regole di composizione degli elementi in frasi eseguibili e costrutti di controllo (istruzioni)
- Semantica: insieme di regole per la rappresentazione degli elementi e delle istruzioni sintatticamente corrette

Descrizione di algoritmi: linguaggio naturale

Un algoritmo viene descritto mediante della azioni elementari che descrivono operazioni semplici e le strutture di controllo che descrivono quali azioni devono essere eseguite e/o in quale ordine.
Esempi: se… allora… altrimenti… finche… esegui… torna al passo…
Questo ha vantaggi e svantaggi: non devo imparare niente di nuovo ma il linguaggio naturale è ambiguo e non è il miglior modo per scrivere algoritmi, quindi dobbiamo usare un linguaggio specifico:

Diagramma a blocchi: grafi di flusso

Studiato apposta per ridurre l'ambiguità del linguaggio naturale, è un linguaggio grafico che si basa su blocchi. Questo linguaggio è composto da simboli grafici. Un particolare simbolo grafico è detto blocco elementare. I blocchi sono collegati tra loro tramite frecce. I blocchi elementari sono sei: un blocco iniziale(blocco da cui parte l'algoritmo, che è uno e uno solo), uno finale(rappresenta la fine, termina l'algoritmo, potrei avere anche più blocchi finali, tutti i percorsi devono terminare con un blocco finale), blocco di lettura(ha una freccia di entrata e una di uscita), blocco di scrittura(scrive il valore della variabile), blocco azione(ha un ingresso e un'uscita, esegue un azione senza ambiguità), blocco di controllo(ha un ingresso e due uscite perché all'interno ha una condizione, ha un uscita sul vero e una sul falso).

Per combinare questi blocchi si seguono delle linee guida che possono essere utilizzate per scrivere l'algoritmo: le strutture di controllo

Strutture di controllo

Sono dei modi per aggregare dei blocchi, esistono tre strutture di controllo:
- sequenza: serie di istruzioni che vengono eseguite una dopo l'altra
- selezione: viene posta una condizione, se è verificata viene eseguito un blocco altrimenti un altro
- ripetizione: uno stesso blocco viene eseguito a ripetizione fino a quando non viene verificata la condizione di uscita

Teorema di Bohm-Jacopini:

I tre costrutti fondamentali(sequenza, selezione, ripetizione) sono sufficienti a descrivere qualunque algoritmo

Considerazioni su diagramma a blocchi:

un diagramma a blocchi è un insieme di blocchi costituito da:
- un blocco iniziale
- un numero finto n>=1 di blocchi azione e/o blocchi di lettura/scrittura
- un numero finito m>=0 di blocchi di controllo
- un blocco finale

Condizioni di validità:
- ciascun blocco è raggiungibile dal blocco iniziale
- il blocco finale è raggiungibile da qualsiasi altro blocco

Schemi a blocchi: variabili

È un contenitore di valori che ha un nome. Una variabile non può essere vuota, cioè ad una variabile è sempre associato un valore

Schemi a blocchi: operatori ed espressioni

-operatori aritmetici: +, -, *, /, … (agiscono sugli operandi e producono un valore numerico)
- operatori di confronto: >, =, <, … (agiscono sugli operandi e producono un valore logico: vero o falso)
- funzioni: cos(x), log(x), … (agiscono su variabili e producono un valore numerico)
- procedure: leggi (x), scrivi (n), … (agiscono su parametri ed effettuano operazioni)
- espressione: 5+cos(y), … (composizione di operatori, funzioni, variabili e costanti. Ad essa è associato un valore)
- assegnamento: X:=6, Y:=X, Z:=X+6, … (il valore dell'espressione a destra dell'operatore di assegnamento è assegnato alla variabile a sinistra)

Pre- e post- condizioni nei cicli

Due algoritmi apparentemente uguali possono in realtà differire semplicemente cambiando l'ordine dei blocchi

Flusso di esecuzione:

Il flusso di controllo definisce tutti i cammini possibili di esecuzione e si basa su ipotesi relative ai valori assumibili dalle variabili

Il tracing è il meccanismo che permette di controllare il funzionamento di un algoritmo

INTRODUZIONE AL C

Il linguaggio C è un linguaggio per scrivere algoritmi. Esistono tantissimi linguaggi di programmazione, il C è ad alto livello, ovvero un linguaggio che permette di scrivere programmi con uno stile vicino al programmatore (ha termini che assomigliano al linguaggio quotidiano). È stato sviluppato all'inizio degli anni '70, è molto diffuso e ha molte applicazioni. È un linguaggio da affrontare con tre passi: elementi, sintassi e semantica. Gli elementi è l'alfabeto/vocabolario del linguaggio, la sintassi è l'insieme di regole con cui si compongono gli elementi per costruire frasi eseguibili e la semantica è il significato degli elementi, delle frasi e dell'intero programma. Le regole sintattiche devono essere univoche sulla composizione delle frasi. Un software chiamato compilatore verifica se il programma è sintatticamente corretto, se questa condizione è verificato viene creato il file eseguibile.

Proprietà del C:

- è un linguaggio dichiarativo, ovvero per poter utilizzare una variabile deve essere definita a priori
- è un linguaggio strutturato, ci mette a disposizione i costrutti base per definire la sequenza, selezione e iterazione (posso scrivere qualunque programma)
- è un linguaggio case sensitive, ovvero differenzia maiuscole e minuscole

Un programma scritto in C è un file di testo strutturato in modo opportuno, viene poi elaborato dal compilatore, controlla che sia stato scritto in modo corretto e genera un file eseguibile. Imparare C significa imparare sia la sintassi che la semantica.

Elementi del linguaggio C:

- Parole chiave (o riservate) proprie del linguaggio
- Identificatori, una sequenza di caratteri che mi permette di indentificare qualche cosa (variabili, costanti, funzioni, ecc…)
- Operatori (unitari o binari), possono essere di assegnamento (=), aritmetici (+, -, *, /), relazionali (>, <, ==, <=, =>, !=) e logici (!, ||)
- separatori di istruzioni (;), commento, espressioni ecc…
- direttive al preprocessore C
- valori costanti (cifre o caratteri)

Esempio di programma in C:

```
/* Questo è il nostro primo programma in C */        (questo è un commento)
#include <stdio.h>                                   (serve per specificare che andremo ad usare delle funzioni)

int main()                                           (ogni programma deve contenere una funzione principale
{                                                    chiamata main)
        printf("Hello world\n");                     (istruzione che stampa a video un messaggio)
        return 0;                                    (serve per terminare la funzione main, termina il programma)
}                                                    (rappresentano il corpo del programma)
```

Funzioni di ingresso e uscita:

in C l'ingresso e l'uscita avviene tramite chiamate di funzioni disponibili nella Standard Library

Output di un programma:

```
/* Programma Room1.c */
#include <stdio.h>

Int main()
{
        printf("La mia stanza è lunga %d metri e larga %d metri\n", 3, 4);
        return 0;
}
```

Il % è uno spazio vuoto che vai a riempire con quello che c'è dopo i due apici e la virgola
Il %d è uno spazio vuoto che vai a riempire con un numero che c'è dopo i due apici e la virgola

Semplici calcoli:

```
#include <stdio.h>

Int main()
{
        printf("la mia stanza è lunga %d metri e larga %d metri\n", 3, 4);
        printf("la mia stanza è grande %d metri quadrati\n", 3*4);
        return 0;
}
```

Le variabili:

Affinché un programma sia in grado di risolvere una classe di problemi bisogna introdurre le variabili

```
#include <stdio.h>

Int main()
{
        Int lung, larg, area;
        lung = 3;
        larg = 4;
        area = lung*larg
        printf("la mia stanza è lunga %d metri e larga %d metri\n", lung, larg);
        printf("la mia stanza è grande %d metri quadrati\n", area);
        return 0;
}
```

Lettura dati dal terminale:

```
#include <stdio.h>

Int main()
{
        Int lung, larg, area;
        scanf("%d", &lung);
        scanf("%d", &larg);
        area = lung*larg;
        printf("la mia stanza è lunga %d metri e larga %d metri\n",lung,larg);
        printf("la mia stanza è grande %d metri quadrati\n",area);
        return 0;
}
```

TIPO DI DATO SEMPLICE

I tipi di dati ci dicono come sono fatte le variabili, è importante inserire le variabili perché una variabile rappresenta uno spazio della memoria RAM, quindi l'esecutore deve sapere quanto spazio serve al programma. Oltre a sapere il numero bisogna anche sapere la tipologia. In C esistono diversi tipi di dato built-in, tra cui:

-int (numeri interi)
- float (numeri con virgola)
- double (numeri con doppia precisione)
- char (caratteri)

Il tipo intero:

consente di rappresentare numeri interi, in C il tipo base per lavorare con gli interi è int

Il tipo reale:

i tipi per lavorare con numeri reali sono il float e il double
per poter leggere un avariabile float devo usare %f