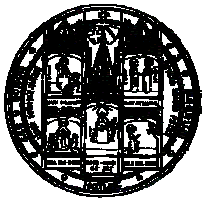


Università degli Studi di Bologna
Facoltà di Ingegneria

Corso di Fondamenti di Informatica L-A

http://lia.deis.unibo.it/Courses/FondA0607-ELE/

Corsi di Laurea in Ingegneria Elettronica e Ingegneria delle Telecomunicazioni

Prof. Paolo Torroni

Anno Accademico 2006-2007

Programma del corso

Elementi di programmazione

1

- Metodi per l'analisi e la risoluzione di un problema. Algoritmi. Rappresentazione degli algoritmi con diagrammi di flusso.
- Metodologia di sviluppo top-down e bottom-up.
- Linguaggi di programmazione. Alfabeto, sintassi e semantica. Formalismo BNF. Fasi di sviluppo di un programma.
- Progetto di una soluzione: modularità, riusabilità, leggibilità del codice, cenni di complessità, scelte ingegneristiche.

Architettura dei sistemi di elaborazione

2

- Hardware e Software. Componenti di un calcolatore elettronico. Gerarchia delle memorie. Funzionamento di una CPU. Funzioni del sistema operativo.
- Architettura astratta di Von Neumann.

Programma del corso

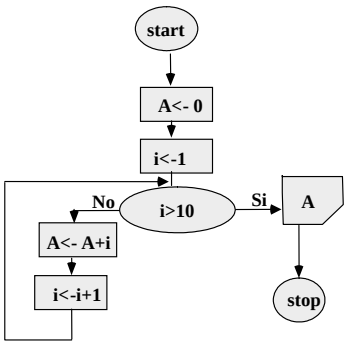
Il linguaggio C

3

- Alfabeto e sintassi del C.
- Tipi di dato scalari e strutturati: vettori, stringhe, record, tabelle, matrici. Puntatori.
- Dichiarazione di costanti, variabili e loro tipo.
- Istruzioni in C. Espressioni. Istruzioni di assegnamento e di ingresso/uscita.
- Istruzioni composte, condizionali e cicli.
- Funzioni e procedure. Dichiarazione/definizione.
- Tecniche di passaggio dei parametri.
- Regole di visibilita` e tempo di vita delle variabili.
- Il modello “run-time” del C. Spazio di indirizzamento. Ricorsione.
- Librerie: stdio.h, stdlib.h e string.h.
- Gestione dei file. Operazioni su file di testo e file binari.
- Modularità in C

1

Informatica
Algoritmi e Strutture dati



1

Cos'è l'INFORMATICA ??

Il termine "*informatica*" ha un' accezione molto ampia.

Esistono varie definizioni:

- l'**informatica** è *la scienza che si occupa della conservazione, dell'elaborazione e della rappresentazione dell'informazione*.
- l'**informatica** *è la scienza che si occupa dello studio dei fondamenti teorici dell'informazione e del calcolo e della loro implementazione e applicazione nei calcolatori*
- *…*
- Scienza dei calcolatori elettronici, scienza dell'informazione, …
- Computer science, information science, …
- …

➔ Definizione proposta nell'ambito di questo corso:

"Scienza della rappresentazione e dell'elaborazione automatica dell'informazione."

1

Informatica

Informazione: tutto ciò che può essere **rappresentato** all'interno di un computer è informazione:

- Numeri
- Caratteri, parole e testi
- Immagini
- Suoni
- Filmati
- comandi (istruzioni) e sequenze di comandi (programmi) che il calcolatore deve eseguire

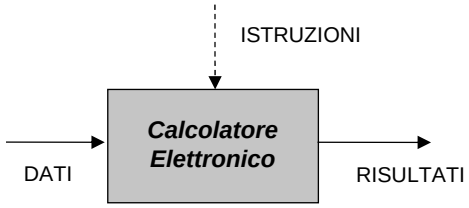
- Le modalità di **rappresentazione** dipendono dalle caratteristiche dell'elaboratore.

Elaboratore Elettronico (computer):
è lo strumento per la rappresentazione e l'elaborazione delle informazioni.

1

Programmazione

- È l'attività con cui si predispone l'elaboratore ad eseguire un particolare insieme di azioni su particolari informazioni (*dati*), allo scopo di risolvere un certo problema.



1

Problemi?

I problemi che siamo interessati a risolvere con l'elaboratore sono di natura molto varia. Ad esempio:

- Semplici operazioni aritmetiche (somma di due numeri interi)
- Applicazioni matematiche (dati a e b, risolvere l'equazione ax+b=0)
- Ordinamento di dati (dato un'insieme di parole, metterle in ordine alfabetico)
- Operazione di ricerca di informazioni (dato un elenco di nomi e relativi numeri di telefono trovare il numero di telefono di una determinata persona)
- Operazioni statistiche (dati gli archivi dei dipendenti di un'azienda, calcolare lo stipendio medio del dipendente dell'azienda)
- Applicazioni multimediali (elaborazione/trasformazione di sequenze di immagini, gaming, enternainment, …)
- Gestioni di grandi quantità di informazioni (gestione di basi di conoscenze, database, datawarehousing, business process…)
- Applicazioni web (portali, customer care, ordini, vendite on-line, news, virtual enterprise…)
- Applicazioni di calcolo scientifico e di progetto industriale (simulazioni, progetto, derivate finanziarie, previsioni metereologiche)

Risoluzione dei Problemi

- La descrizione del problema non fornisce (in genere) un metodo per calcolare il risultato.
- Non tutti i problemi sono risolvibili attraverso l'uso del calcolatore. In particolare esistono classi di problemi per le quali la soluzione automatica non e` proponibile.
- **Ad esempio:**
  - se il problema presenta infinite soluzioni
  - per alcuni dei problemi **non è stato trovato** un metodo risolutivo.
  - per alcuni problemi e'stato dimostrato che **non esiste** un metodo risolutivo automatizzabile

Noi ci concentreremo sui problemi che, ragionevolmente, ammettono un metodo risolutivo (esprimibile mediante una ***funzione calcolabile***).

Risoluzione di un problema

Con questo termine si indica il processo che:

- dato un problema, e
- individuato un metodo risolutivo

trasforma i dati iniziali nei corrispondenti risultati finali.

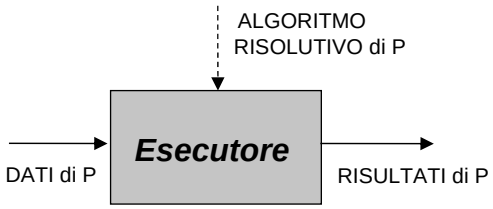
- Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso del calcolatore, tale processo deve poter essere definito come sequenza di **azioni elementari**, esprimibili mediante **istruzioni**.

1

ALGORITMO

È **l'insieme ordinato** delle azioni che risolve un dato problema P.

- l'algoritmo descrive un metodo risolutivo attraverso un insieme ordinato di azioni.
- l'esecuzione dell'algoritmo è affidata ad un generico "***esecutore***", cioè una macchina astratta (non necessariamente un calcolatore !) in grado di interpretare ed eseguire ogni azione specificata nell'ordine indicato.



1

Esecutore e istruzioni primitive

- Ad un generico esecutore è implicitamente associato un insieme di istruzioni primitive (set di istruzioni):
  - sono le sole istruzioni che è in grado di interpretare ed eseguire.

1

Esempio: la preparazione del caffè

Esecutore: essere umano corredato di caffettiera "moka", cucina a gas e macina-caffè;

Algoritmo:

1. **svitare** la caffettiera;
2. se si dispone di caffè macinato:
   - **riempire** il filtro con il caffè macinato,
   - altrimenti se si dispone di caffè in chicchi:
     - **macinarlo** e ripetere il punto 2;
     - altrimenti **terminare** (il caffè non si può fare..).
3. **riempire** la parte inferiore della caffettiera con acqua;
4. **inserire** il filtro nella macchina;
5. **avvitare** la caffettiera;
6. **accendere** il fuoco a gas;
7. **collocare** la moka sul fuoco;
8. **attendere** l'uscita del caffè;
9. **spegnere** il fuoco;
10. **fine** (il caffè è pronto).

1

Esempio: la preparazione del caffè

Esecutore: essere umano corredato di caffettiera "moka", cucina a gas e macina-caffè;

Set di istruzioni:

- operazioni fondamentali sulla caffettiera:
  - svitare
  - avvitare
  - riempire il filtro
  - riempire con acqua
- operazioni fondamentali sulla cucina a gas:
  - accendere
  - spegnere
- operazioni fondamentali sul macina-caffè:
  - macinare
- altre operazioni:
  - verifica di condizioni
  - ripetizione di operazioni
  - attesa
  - ..

Fondamenti di Informatica L- A

1

Proprietà fondamentali dell'Algoritmo

1. **Eseguibilità:** ogni "istruzione" deve essere eseguibile da parte dell'esecutore dell'algoritmo;
2. **Non Ambiguità:** ogni istruzione deve essere univocamente interpretabile dall'esecutore
3. **Finitezza:** il numero totale di azioni da eseguire, per ogni insieme di dati di ingresso, e' finito.

➔ **se almeno una delle 3 proprietà non è soddisfatta, la sequenza non è un algoritmo.**

Altre proprietà desiderabili:

- **generalità**: corretto funzionamento dell'algoritmo anche variando alcuni aspetti del problema (ad esempio, la dimensione dell'insieme dei dati, il tipo dei dati, ecc.)
- **efficienza**: tanto minore è il numero di azioni eseguite per la risoluzione del problema, tanto maggiore è l'efficienza.
- **determinismo**: possibilità di prevedere esattamente prima dell'esecuzione la sequenza di azioni che verranno eseguite, per ogni insieme di dati.
- ...

1

Algoritmi e Programmi

- Se l'esecutore è un **elaboratore elettronico**:
  1. è necessario conoscere **l'insieme di istruzioni** che è in grado di interpretare/eseguire
  2. è necessario conoscere quali **tipi di informazione** (dati) è in grado di rappresentare

Gli aspetti 1. e 2. sono peculiari del **formalismo** scelto per esprimere l'algoritmo all'interno del sistema di elaborazione, cioè del

Linguaggio di Programmazione

Fondamenti di Informatica L- A

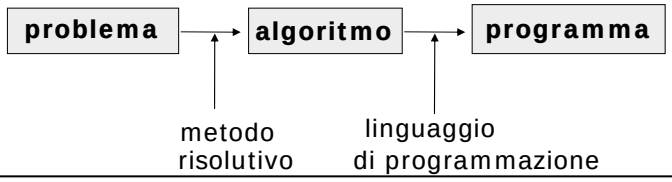
1

Algoritmi e Programmi

Quindi:
Dato un problema P, la sua soluzione può essere ottenuta mediante l'uso del calcolatore, compiendo i seguenti passi:

1. individuazione di un **metodo risolutivo**
2. scomposizione del procedimento in insieme ordinato di azioni: **algoritmo**
3. rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile per l'elaboratore (il linguaggio di programmazione).

Si ottiene così il **PROGRAMMA**, che potrà essere eseguito dall'elaboratore per risolvere automaticamente ogni istanza del problema P.



1

Algoritmi equivalenti

Due algoritmi si dicono equivalenti quando:

- hanno lo stesso dominio dei dati (dominio di ingresso);
- hanno lo stesso dominio dei risultati (dominio di uscita);
- in corrispondenza degli stessi valori nel dominio di ingresso producono gli stessi valori nel dominio di uscita

Due algoritmi equivalenti:

- forniscono lo stesso risultato
- possono essere profondamente diversi
- possono avere differente efficienza

1

Algoritmi Equivalenti: Calcolo del massimo comun divisore

Dati due interi m ed n, calcolare il massimo comune divisore di essi.

Algoritmo a:

1. Calcola l'insieme I dei divisori di m
2. Calcola l'insieme J dei divisori di n
3. Calcola l'insieme K dei divisori comuni:
   $K = I \cap J$
4. Calcola il massimo in K: questo e' il risultato

1

Algoritmi Equivalenti: Calcolo del massimo comun divisore

Algoritmo b: si basa sul *metodo di Euclide:* detta *mcd* la funzione che calcola la soluzione del problema, la sua definizione è data come segue:

- mcd(m,n) = m (oppure n) se m=n
- mcd(m,n) = mcd(m-n, n) se m>n
- mcd(m,n) = mcd(m, n-m) se m<n

Quindi l'algoritmo b si può esprimere così:

1. Finché m è diverso da n ripeti le seguenti azioni:
   - se m>n sostituisci a m il valore (m-n)
   - altrimenti sostituisci a n il valore (n-m)
2. Il massimo comun divisore è n

➢ **Gli algoritmi a e b sono equivalenti.**

Fondamenti di Informatica L- A

1

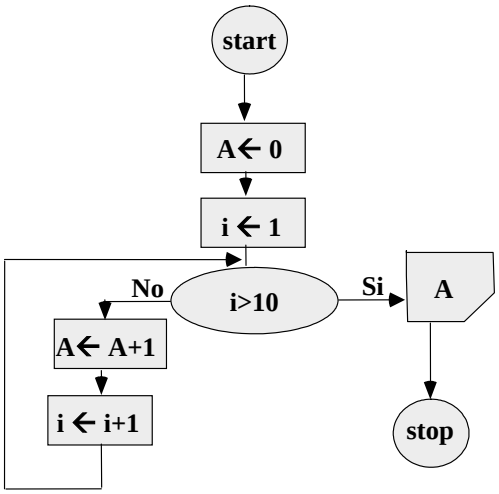
Rappresentazione di Algoritmi: Diagrammi di flusso

E' un formalismo che consente di rappresentare graficamente gli algoritmi.

- un **diagramma di flusso** descrive le azioni da eseguire ed il loro ordine di esecuzione.
- ad ogni tipo di **azione** corrisponde un simbolo grafico (**blocco**) diverso.
- ogni blocco ha un ramo in ingresso ed uno o piu` rami in uscita; collegando tra loro i vari blocchi attraverso i rami, si ottiene un diagramma di flusso
- un diagramma di flusso appare, quindi, come un insieme di blocchi, collegati fra loro da linee orientate che specificano la sequenza in cui i blocchi devono essere eseguiti (flusso del controllo di esecuzione).

1

Esempio



1

Diagrammi di Flusso

- **Dati:**
  - **Variabile**: Rappresenta un dato ed è individuata da un nome simbolico cui è assegnato un valore che può cambiare durante l'esecuzione dell'algoritmo.
  - **Costante**: è una grandezza nota a priori, il cui valore non cambia durante l'esecuzione.
- **Blocco (o istruzione):** rappresenta una operazione mediante un simbolo grafico
  - **Blocco semplice**: esecuzione di una singola operazione elementare sui dati
  - **Blocco condizione**: in base al verificarsi di una condizione, permette di differenziare il comportamento dell'algoritmo, mediante la scelta tra due alternative.

1

Diagrammi di Flusso

- **Espressioni**: sequenze di variabili e costanti combinate fra loro mediante **operatori**
  - operatori aritmetici: ad esempio {+, -, *, /}:
    s + 5 ➔ producono un risultato *aritmetico*
  - operatori logici e relazionali: ad esempio {and, or, not} e {<, >, =, ≤, ≥, ≠}
    not (C > B) ➔ producono un risultato *logico* {*vero, falso*}

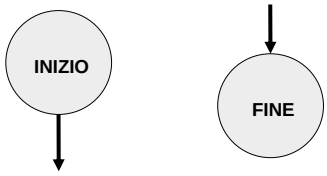
Fondamenti di Informatica L- A

1

Blocchi semplici

Inizio e fine esecuzione (start e stop)

- Marcano inizio e fine di un algoritmo
- Inizio è il blocco da cui deve iniziare l'esecuzione (uno solo per ogni algoritmo).
- Il blocco fine fa terminare l'esecuzione dell'algoritmo (almeno uno).



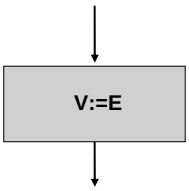
1

Blocchi semplici

Assegnamento

calcola il valore dell'espressione a destra del simbolo ":=" (o "←") e lo si attribuisce (lo si *assegna*) alla variabile indicata a sinistra del simbolo (con eventuale perdita del valore precedente di V)

Esempio:



dove V è il nome di una variabile, E è una espressione.

Significato: *" Calcola il valore dell'espressione E e assegnalo alla variabile V."*

N.B. Il valore di V viene, in generale, **modificato.**

Fondamenti di Informatica L- A

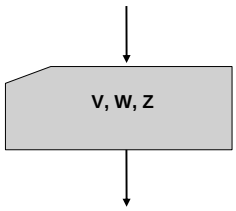
1

Blocchi semplici

Ingresso (lettura, read, input):

Si ricevono dal dispositivo di ingresso (per esempio, la tastiera) tanti valori quante sono le variabili specificate all'interno del blocco (separate da virgole), e si attribuiscono (*si assegnano*) nello stesso ordine alle variabili.

Ad esempio:



Significato: "*leggi i tre valori dati in ingresso, ed assegnali rispettivamente alle variabili V, W, e Z*."

Quindi: durante l'esecuzione, se vengono digitati dalla tastiera i valori: 5, 7, 9, allora la variabile V assumerà il valore 5, W il valore 7 e Z il valore 9.

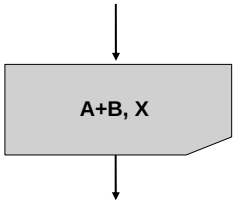
1

Blocchi semplici

Uscita (stampa, print, output):

i valori delle espressioni specificati all'interno del blocco vengono calcolati e successivamente trasmessi al dispositivo di uscita (per esempio, il video).

Ad esempio:

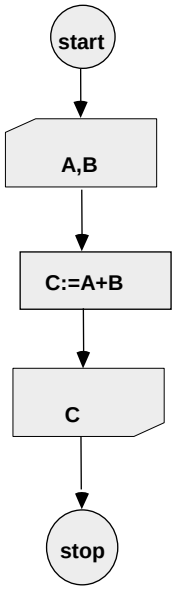


Significato: "*calcola il valore dell'espressione A+B e di X e trasmettili in uscita*."

Quindi: se A vale 10, B vale 7 e X vale -25, l'esecuzione del blocco causerà la stampa dei 2 valori: 17 e -25.

NB: I valori di A, B e X non vengono alterati dall'esecuzione del blocco.

Esempio

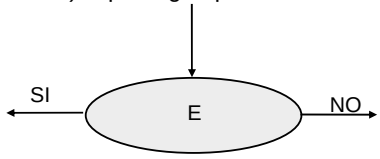


Blocco Condizione

Condizione:

Si valuta la condizione specificata all'interno del blocco: se è verificata, si prosegue con la linea di flusso contrassegnata da "*SI*" (o *ok*, *vero*, *true*..), altrimenti (se non e` verificata) si prosegue per il ramo etichettato con "NO" (*falso*, *false*,..).

Esempio:



dove E è un'espressione relazionale (o logica): ritorna valore vero, oppure falso.

Significato: " *Calcola il valore dell'espressione E: se e` vero, prosegui per il ramo SI, altrimenti prosegui per il ramo NO*".

NB. Il blocco condizione è l'elemento di base per realizzare *alternative* e *ripetizioni*.

1

Strutture di controllo

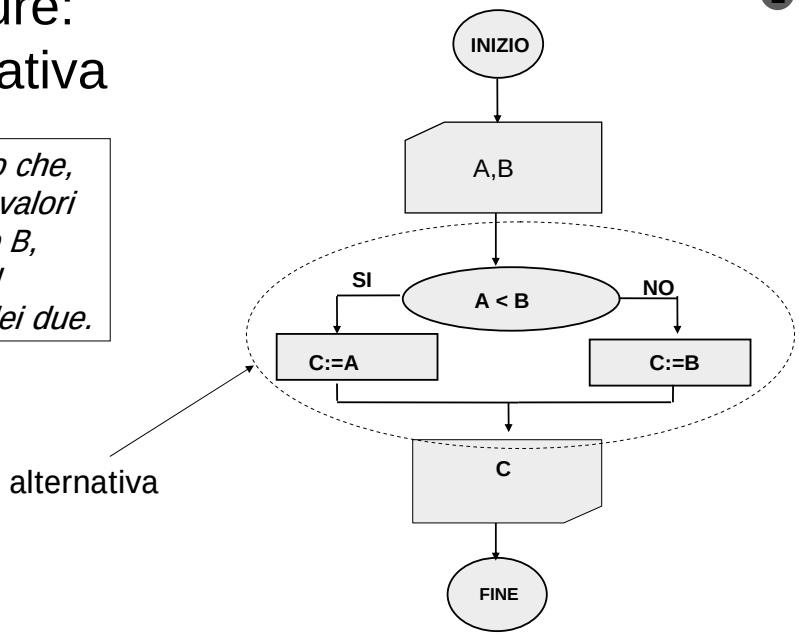
Mediante i blocchi fondamentali finora visti, è possibile costruire delle strutture da utilizzare per il controllo del flusso di esecuzione dell'algoritmo:

- **Alternativa**:esprime la **scelta tra due possibili azioni** (o sequenze di azioni) mutuamente esclusive.
- **Ripetizione**: **esprime la ripetizione di una sequenza di istruzioni.**

1

Strutture: Alternativa

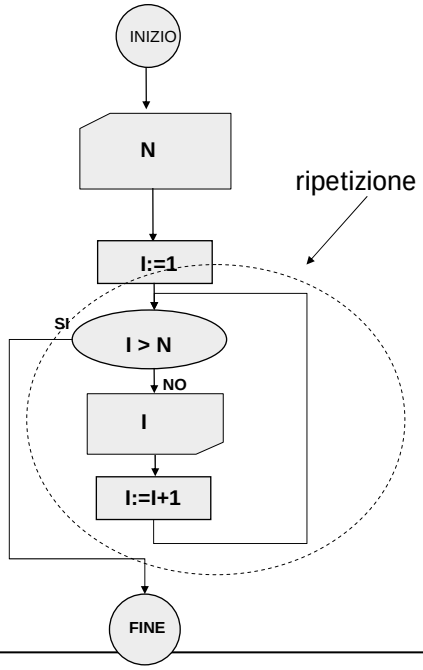
algoritmo che, dati due valori interi A e B, stampa il minore dei due.



alternativa

Strutture: ripetizione

algoritmo che, dato un valore intero positivo N, stampa tutti gli interi >0 e <= N.

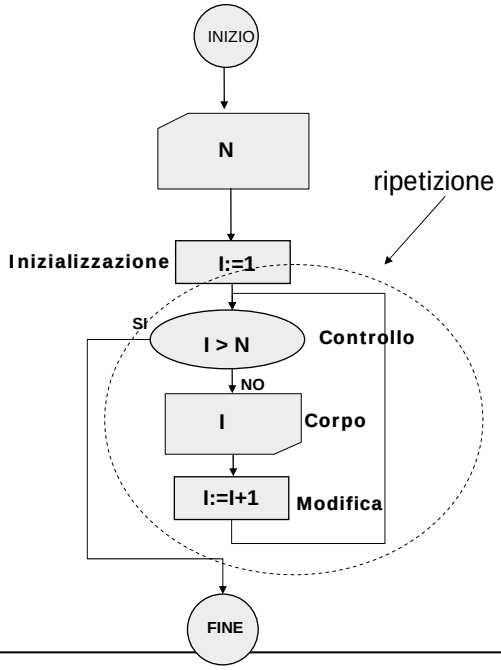


Strutture: Ripetizione (o *iterazione*)

Nel caso piu` generale, e` costituita da 4 elementi:

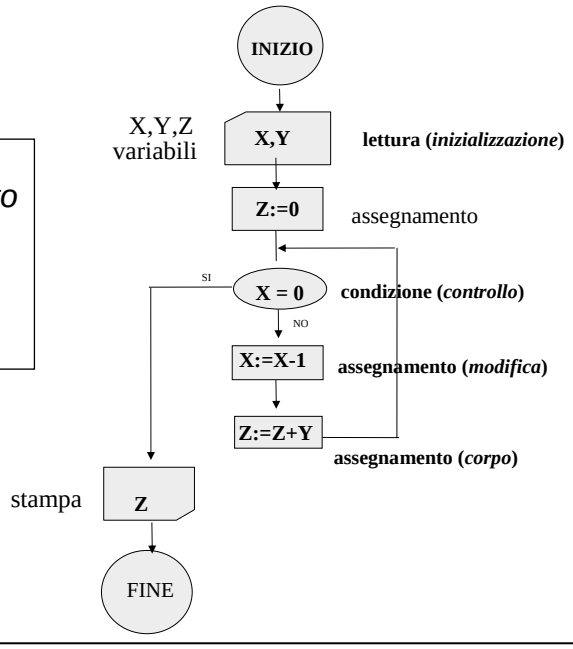
- **Inizializzazione**: assegnazione dei valori iniziali alle variabili caratteristiche del ciclo (viene eseguita una sola volta);
- **Corpo**: esecuzione delle istruzioni fondamentali del ciclo che devono essere eseguite in modo ripetitivo;
- **Modifica**: modifica dei valori delle variabili che controllano l'esecuzione del ciclo (eseguito ad ogni iterazione);
- **Controllo**: determina, in base al valore delle variabili che controllano l'esecuzione del ciclo se il ciclo deve essere ripetuto o meno.

Ripetizione



Esempio:

Algoritmo che calcola il prodotto come sequenza di somme (si suppone X ≥ 0).



2

Elaboratore Elettronico ("*computer*")

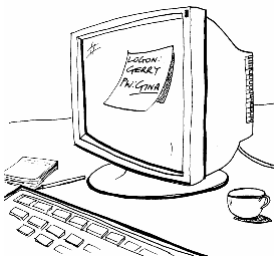
computer =
"strumento per la rappresentazione e l'elaborazione delle informazioni"

2

L'ELABORATORE

Componenti principali
- Unità centrale (Case)
  - •Motherboard + memoria, bus, schede, …
  - •Lettore CD/DVD/floppy
  - •Dischi fissi ("hard disk")
  - •Porte USB/RJ-45/…
- Video ("monitor")
- Tastiera e Mouse

Componenti accessori
- Stampante
- Modem
- Scanner, tavolette grafiche
- …

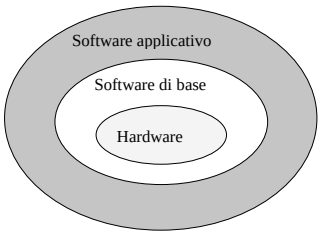


HARDWARE

2

SOFTWARE

Software: programmi che vengono eseguiti dal sistema.



Distinzione fra:

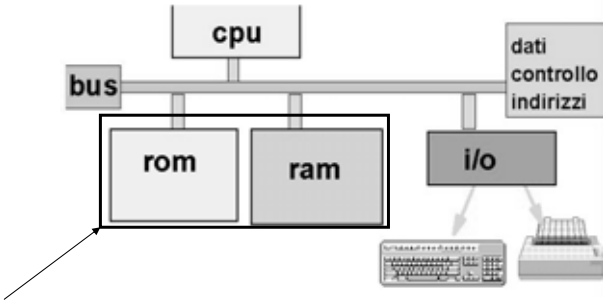
- *Software di base* (es. Sistema Operativo)
- *Software applicativo*

NB: suddivisione utile ma non sempre evidente (*firmware*)

2

HARDWARE

E' composto da un insieme di ***unità funzionali***



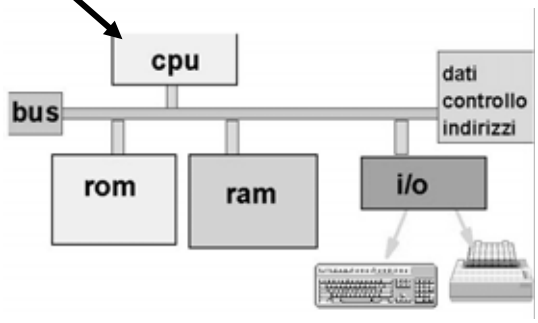
memoria centrale

2

HARDWARE

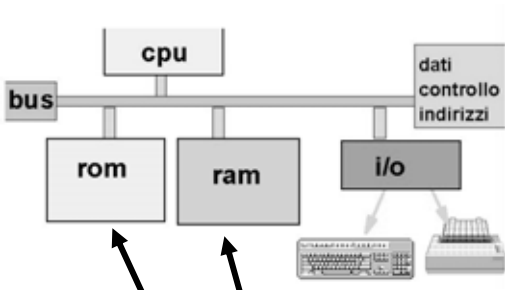
CPU (Central Processing Unit), o **Processore**

CPU: Svolge le elaborazioni e il trasferimento dei dati, cioè *esegue i programmi*



2

HARDWARE



RAM & ROM

- Dimensioni relativamente limitate
- Accesso molto rapido

RAM (*Random Access Memory*), e
ROM (*Read Only Memory*)
Insieme formano la *Memoria centrale*

2

HARDWARE

ATTENZIONE

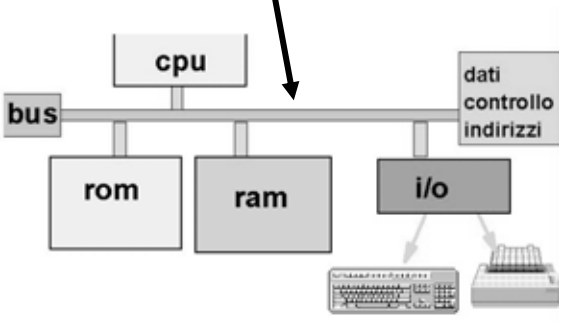
- **RAM è volatile** (perde il suo contenuto quando si spegne il calcolatore)
  - usata per memorizzare dati e programmi
- **ROM è persistente** (mantiene il suo contenuto quando si spegne il calcolatore) ma il suo ***contenuto è fisso e immutabile***
  - usata per memorizzare programmi di sistema

2

HARDWARE

BUS DI SISTEMA

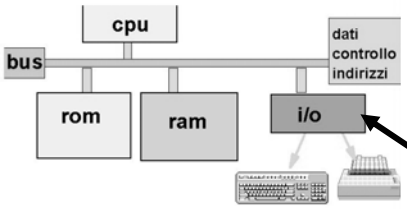
È una “linea di comunicazione” che collega tutti gli elementi funzionali.



Fondamenti di Informatica L- A

2

HARDWARE



Sono i dispositivi (*periferiche*) usati per far comunicare il calcolatore con l'esterno (in partico-lare con l'utente)

UNITÀ DI INGRESSO / USCITA (I/O)

- Tastiera e Mouse
- Video e Stampante
- Scanner
- Tavoletta grafica
- **Dispositivi di memoria di massa**
- ...

2

TECNOLOGIA DIGITALE

CPU, memoria centrale e dispositivi sono realizzati con **tecnologia elettronica digitale**.

Dati ed operazioni vengono codificati a partire da due valori distinti di grandezze elettriche:

- tensione alta (V_H, 5V)
- tensione bassa (V_L, 0V)

A tali valori vengono convenzionalmente **associate le due cifre binarie 0 e 1:**

- **logica positiva: $1 \leftrightarrow V_H$, $0 \leftrightarrow V_L$**
- logica negativa: $0 \leftrightarrow V_H$, $1 \leftrightarrow V_L$

2

TECNOLOGIA DIGITALE (segue)

Dati ed operazioni vengono codificati tramite **sequenze di bit**

01000110101 ....

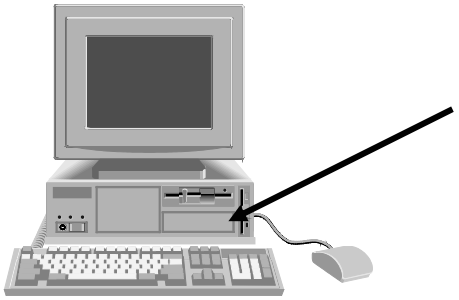
CPU è in grado di operare soltanto in aritmetica binaria, effettuando operazioni *elementari* :

- somma e differenza
- scorrimento (shift)
- ...

Lavorando direttamente sull'hardware, **l'utente è forzato a esprimere i propri comandi *al livello della macchina*, tramite *sequenze di bit.***

2

MEMORIA DI MASSA



- Dischi
- CD, DVD, …
- Nastri
- ...

- memorizza **grandi quantità** di informazioni
- **persistente** (le informazioni sopravvivono all'esecuzione dei programmi: non si perdono spegnendo la macchina)
- accesso molto meno rapido rispetto alla memoria centrale (**milli**secondi contro **nano**secondi / differenza 10^6)

2

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAMENTALI:

- **ad *accesso sequenziale*** (ad esempio, **NASTRI**): per recuperare un dato è necessario accedere prima a tutti quelli che lo precedono sul dispositivo
- **ad *accesso diretto*** (ad esempio, **DISCHI e penne USB**): si può recuperare direttamente un qualunque dato memorizzato

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

2

DISPOSITIVI MAGNETICI

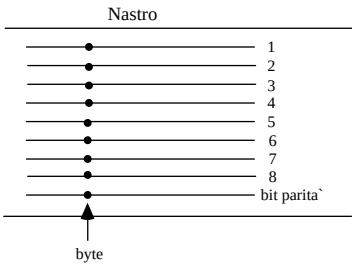
- L'area del dispositivo è suddivisa in **micro-zone**
- Ogni micro-zona memorizza una **informazione elementare** sotto forma **di *stato di magnetizzazione*:**
  area magnetizzata / area non magnetizzata
- Ai due possibili stati di magnetizzazione vengono **associate le due cifre binarie 0 e 1**
  bit (Binary digIT)
- Quindi, **ogni micro-zona memorizza 1 bit**
- Per memorizzare informazioni più complesse si considerano *collezioni di bit*:
  BYTE (collezione di **8 bit**) e suoi multipli

2

NASTRI MAGNETICI

Nastri di materiale magnetizzabile arrotolati su supporti circolari, o in cassette.

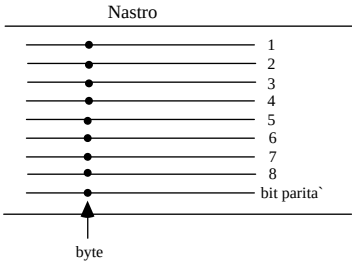
Sul nastro sono tracciate delle **piste orizzontali parallele** (di solito 9, di cui 8 corrispondono ad un byte e la nona è il bit di parità).



2

NASTRI MAGNETICI

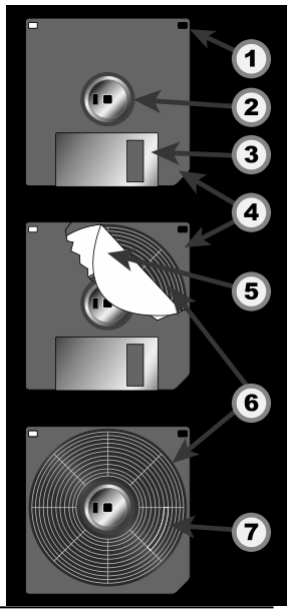
I dati sul nastro sono organizzati in zone contigue dette ***record***, separate da zone prive di informazione (*inter-record gap*).



- Tutte le **elaborazioni** sono **sequenziali:** le operazioni su uno specifico record sono ***lente***
- Oggi servono solo per mantenere copie di riserva (***backup***) dei dati

2

DISCHETTI (FLOPPY)



Fine anni ’60-oggi (poco)

Sono dischi magnetici di ***piccola capacità***, portatili, usati per trasferire informazioni tra computer diversi.

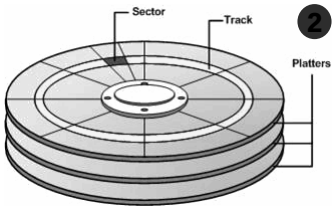
Sono costituiti da un **unico disco** con due superfici.

Sopravvivono solo quelli da 3.5” di diametro (1.4 MB)

Dati scritti su *tracce* (7), create in fase di ***formattazione***.

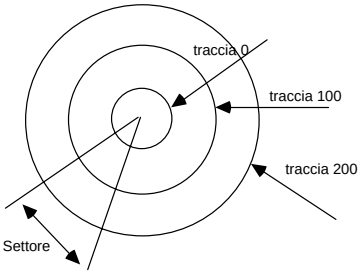
2

DISCHI MAGNETICI



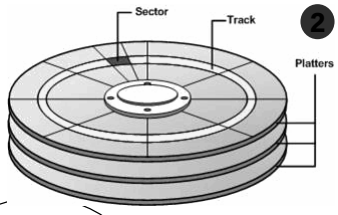
Un disco consiste in un certo numero di ***piatti*** con **due superfici** che ruotano attorno ad un perno centrale.

Ogni superficie dispone di una propria **testina di lettura / scrittura.**

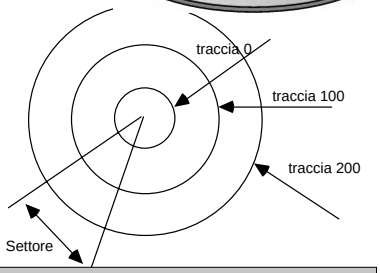


Le superfici sono organizzate in ***cerchi concentrici*** (**tracce**) e in ***spicchi*** di ugual grandezza (**settori**).
Le tracce equidistanti dal centro formano un **cilindro.**

DISCHI MAGNETICI



I dati sono scritti in posizioni successive *lungo le tracce*: ogni bit corrisponde a uno stato di *magnetizzazione* del materiale magnetico della superficie del disco.



Ogni ***blocco*** del disco è identificato con la terna
⟨***superficie***, ***traccia***, ***settore***⟩
Per effettuare il trasferimento dei dati in memoria centrale occorre disporre di un'area di memoria (*buffer*) di dimensioni pari al blocco.

2

DISCHI MAGNETICI: PERFORMANCE

Grandezze da tenere in considerazione:
seek time (~10 ms)
vel. rotazione (~10000 RPM)
transfer rate (~ 10 MBps)

Tempo medio necessario a trasferire un blocco di dati da HD a RAM: (~)
seek time + ½ rotation time +
dimensione dati / transfer rate
e.g. blocco da 1MB:
10ms + 3ms + 1MB / 10 MBps
= 13 ms (latency) + 100 ms (transfer)

2

DISPOSITIVI OTTICI

1984, CD-ROM (Compact-Disk Read-Only Memory)
- Capacità: > 600 MB
- Costo: < $1
- Velocità di trasferimento:
  - originariamente 150 KB / s (“1X”)
  - oggi 24, 32, 52 volte tanto…

1984, WORM (Write Once Read Many)
- Sono dischi ottici scrivibili (una sola volta)
- Parenti stretti dei CD audio (CD-DA, 1982)
- Accesso diretto ai settori (capacità 2.048 KB)

2

DISPOSITIVI OTTICI

1986, CD - I (Compact-Disk Interactive)
- Per memorizzare immagini, filmati, grafica, suono, testi e dati (*multimedialità*).

1997, DVD (Digital Video Disk)
- Evoluzione del CD-ROM
- Capacità di 5,7 / 20 / 50 … GB
- Velocità di trasferimento molto elevata

DVD inizialmente ideato per film e opere pesantemente multimediali.

Fondamenti di Informatica L- A

DISPOSITIVI OTTICI

Ormai il CD/DVD sono (assieme alla rete) tra i principali mezzi per lo scambio di grandi quantità di informazioni

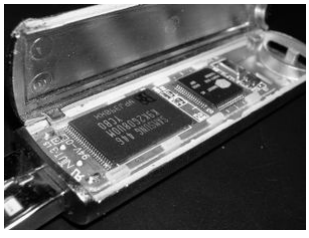
- installazione di nuovi programmi di utilità
- archiviazione di immagini, suoni, opere multimediali
- copie di riserva (backup)
- distribuzione di materiale pubblicitario o “di prova”

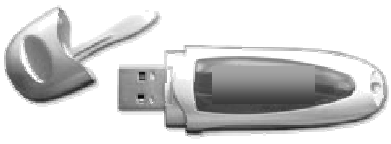
Affidabilità: fino a 10-15 anni.

USB STICK

Flash memory stick ("penne USB"):

- memorie persistenti che possono essere riscritte/cancellate piu` volte
- capacità: ordine dei GB
- Nota: USB 1.1 vs. 2.0: 40 volte più veloce





2

CAPACITÀ DELLE MEMORIE

Tipo di memoria	Capacità
Memoria centrale	10^2 MB – GB
Dischi magnetici	10 GB - TB
Penne USB	10^2 MB - GB
Nastri (bobina)	10 - 10^2 MB
Nastri (cassetta)	10^2 MB - 10 GB
Dischi ottici	650 MB - 50 GB

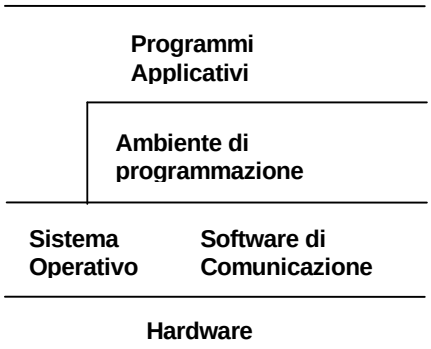
2

IL SOFTWARE

Software:
insieme di programmi eseguibili dal computer.

Organizzazione a strati, ciascuno con funzionalità di livello più alto rispetto a quelli sottostanti

Concetto di ***macchina virtuale***



2

IL FIRMWARE

Firmware:
il confine fra hardware e software.

È uno strato di *micro-programmi*, *scritti dai costruttori*, che agiscono direttamente al di sopra dello strato hardware

Sono memorizzati su una speciale *memoria centrale permanente* (ROM, EPROM, …)

2

IL SISTEMA OPERATIVO

Strato di programmi che opera *al di sopra di hardware e firmware* e **gestisce l'elaboratore**.

Solitamente, è venduto insieme all'elaboratore.

Per lo stesso elaboratore, spesso **si può scegliere tra *diversi sistemi operativi***, con diverse caratteristiche.

Esempi:

- Windows (2000, NT, XP...)
- Unix (BSD, Solaris, …)
- Linux (varie distribuzioni)
- MacOS

2

FUNZIONI DEL SISTEMA OPERATIVO

Le funzioni messe a disposizione dal S.O. dipendono dalla complessità del sistema di elaborazione:

- interpretazione ed esecuzione di comandi
- gestione delle risorse disponibili:cpu, dispositivi, ecc.
  - gestione della memoria centrale
  - organizzazione e gestione della memoria di massa
- gestione di un sistema multi-utente:
  - concorrenza delle attivita`
  - protezione
  - una macchina astratta (o virtuale) per ogni utente:

Un utente "vede" l'elaboratore solo tramite il Sistema Operativo
→ il S.O. realizza una "macchina virtuale"

2

FUNZIONI DEL SISTEMA OPERATIVO

Conseguenza:
diversi S.O. possono realizzare *diverse macchine virtuali* **sullo stesso elaboratore fisico**

Interazione con l'utente:
Attraverso il S.O. il livello di interazione fra utente ed elaboratore viene elevato:

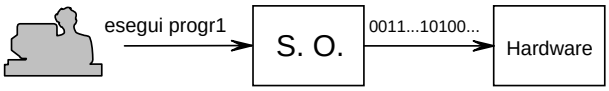
- senza S.O.: sequenze di bit
- con S.O.: comandi, programmi, dati

Fondamenti di Informatica L- A

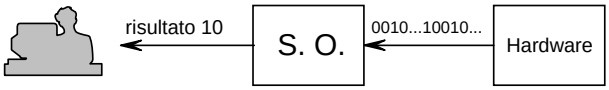
2

RUOLO DEL SISTEMA OPERATIVO

Il S.O. **traduce le richieste dell'utente** in opportune **sequenze di istruzioni,** a loro volta trasformate in **valori e impulsi elettrici** per la macchina fisica.



e viceversa:



2

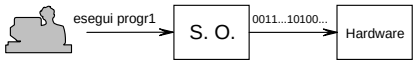
RUOLO DEL SISTEMA OPERATIVO

Qualsiasi operazione di accesso a risorse della macchina implicitamente richiesta dal comando di utente **viene esplicitata dal S.O.**

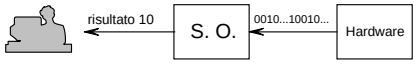
Esempi:
- accesso a memoria centrale
- accesso ai dischi
- I/O verso video, tastiera, …

2

ESEMPIO



e viceversa:



Utente:	Sistema Operativo:
"esegui progr1"	- input da tastiera - ricerca codice di "progr1" su disco - carica in memoria centrale codice e dati <elaborazione>
Utente:	Sistema Operativo:
"stampa 10"	- output su video

2

CLASSIFICAZIONE dei S.O.

In base al numero di utenti:

- **Mono-utente (*mono-user*):** un solo utente alla volta può utilizzare il sistema
- **Multi-utente (*multi-user*):** più utenti possono interagire contemporaneamente con la macchina.

Nel caso di più utenti contemporanei, **il Sistema Operativo deve fornire a ciascuno l'astrazione di *un sistema "dedicato".***

2

CLASSIFICAZIONE dei S.O.

In base al numero di programmi in esecuzione:

- **Mono-programmato (*mono-task*):** si può eseguire *un solo programma* per volta
- **Multi-programmato (*multi-task*):** il S.O. è in grado di portare avanti contemporaneamente l'esecuzione di più programmi (pur usando una sola CPU).

Nel caso di multi-programmazione **il S.O. deve gestire la suddivisione del tempo** della CPU fra i vari programmi.

2

CLASSIFICAZIONE dei S.O.

Esempi:

- **MS-DOS**: monoutente, monoprogrammato
- **Windows95/98**: monoutente, multiprogrammato
- **Windows XP**: mono/multiutente,multiprogrammato
- **UNIX e Linux**: multiutente, multiprogrammato

2

PROGRAMMI APPLICATIVI

Risolvono problemi specifici degli utenti:

- *word processor*: elaborazione di testi
- *fogli elettronici*: gestione di tabelle, calcoli e grafici
- *database*: gestione di archivi
- *suite* (integrati): collezione di applicativi capaci di funzionare in modo integrato come un'applicazione unica.

- Sono scritti in **linguaggi di programmazione** di alto livello
- Risentono in misura ridotta delle caratteristiche della architettura dell'ambiente sottostante (*portabilità*)

2

AMBIENTI DI PROGRAMMAZIONE

È l'insieme dei programmi che consentono la scrittura, la verifica e l'esecuzione di nuovi programmi (***fasi di sviluppo***).

Sviluppo di un programma:

- Affinché un programma scritto in un qualsiasi linguaggio di programmazione sia comprensibile (e quindi eseguibile) da un calcolatore, *occorre* ***tradurlo*** dal linguaggio originario al linguaggio della macchina.
- Questa operazione viene normalmente svolta da speciali programmi, detti ***traduttori***.

2

TRADUZIONE DI UN PROGRAMMA

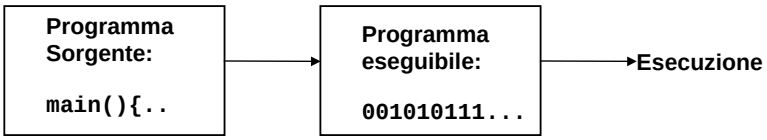
PROGRAMMA	TRADUZIONE
`main()`	
`{ int A;`	`00100101`
`...`	
`A=A+1;`	`11001..`
`if....`	`1011100..`

Il traduttore converte

- ***il testo*** di un programma scritto in un particolare linguaggio di programmazione (***sorgenti***)
- nella corrispondente ***rappresentazione in linguaggio macchina*** (programma ***eseguibile***).

2

SVILUPPO DI PROGRAMMI

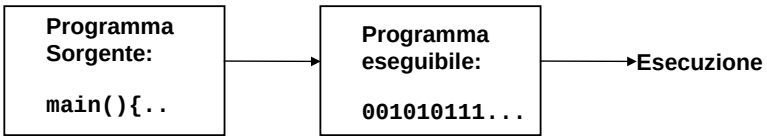


Due categorie di traduttori:

- i ***Compilatori*** traducono l'intero programma (senza eseguirlo!) e producono in uscita il programma convertito in linguaggio macchina
- gli ***Interpreti*** traducono ed eseguono immediatamente **ogni singola istruzione** del *programma sorgente.*

2

SVILUPPO DI PROGRAMMI

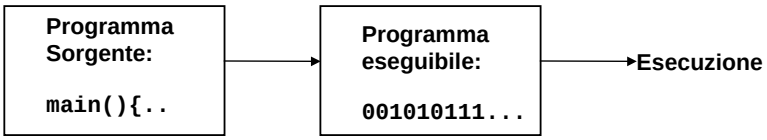


Quindi:

- **nel caso del compilatore**, lo schema precedente viene percorso ***una volta sola*** prima dell'esecuzione
- **nel caso dell'interprete**, lo schema viene invece attraversato ***tante volte quante sono le istruzioni*** che compongono il programma.

2

SVILUPPO DI PROGRAMMI



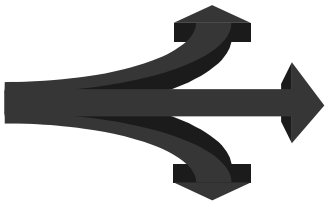
L'esecuzione di un programma ***compilato*** è **più veloce** dell'esecuzione di un programma ***interpretato***

AMBIENTI DI PROGRAMMAZIONE

COMPONENTI

- **Editor:** serve per creare file che contengono **testi** (cioè sequenze di caratteri).
  In particolare, l'editor **consente di scrivere il *programma sorgente.***

E poi….



AMBIENTI DI PROGRAMMAZIONE

I° CASO: COMPILAZIONE

- **Compilatore:** opera la **traduzione di un programma *sorgente*** (scritto in un linguaggio ad alto livello) **in un *programma oggetto*** direttamente eseguibile dal calcolatore.

PRIMA si traduce ***tutto il programma***
POI si esegue **la versione tradotta**.

2

AMBIENTI DI PROGRAMMAZIONE

I° CASO: COMPILAZIONE (segue)

- **Linker:** (*collegatore*) nel caso in cui la costruzione del programma oggetto richieda l'unione di ***più moduli*** (compilati separatamente), il linker provvede a **collegarli** formando un unico *programma eseguibile**.***
- **Debugger:** (*"spulciatore"*) consente di **eseguire passo-passo** un programma, **controllando via via quel che succede**, al fine di ***scoprire ed eliminare errori*** non rilevati in fase di compilazione.

2

AMBIENTI DI PROGRAMMAZIONE

II° CASO: INTERPRETAZIONE

- **Interprete:** ***traduce ed esegue*** direttamente ***ciascuna istruzione*** del *programma sorgente*, **istruzione per istruzione**.
  È alternativo al compilatore (raramente sono presenti entrambi).

Traduzione ed esecuzione sono *intercalate*, e avvengono *istruzione per istruzione.*

Fondamenti di Informatica L- A

2

PERSONAL COMPUTER

PC (ex “IBM-COMPATIBILI”)

Usano processori della famiglia *Intel 80x86*:

- 8086, 386, 486, …
- …
- Intel Pentium, Centrino, Xeon, …
- AMD 64 Athlon, Celeron, Duron, …
- ...

Le prestazioni dipendono in gran parte da:
- frequenza dell’orologio di sistema (*clock*)
- dimensione della RAM
- velocità/parallelismo delle linee dati/comandi (bus)

2

ALTRI SISTEMI DI CALCOLO

Workstation
sistemi con capacità di supportare più attività contemporanee, spesso dedicati a più utenti. Prestazioni normalmente superiori a quello di un tipico Personal Computer.

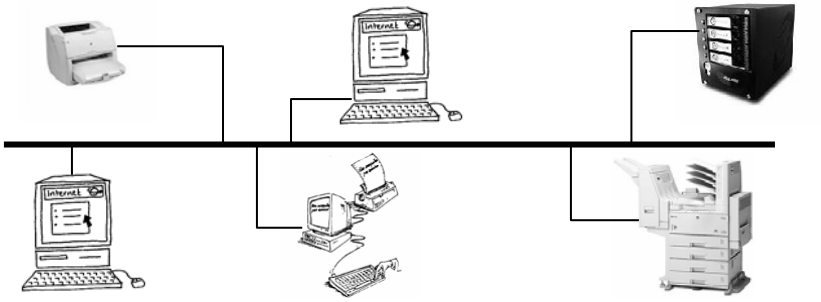
Mini-calcolatori
Macchine capaci di servire decine di utenti contemporaneamente, collegati tramite terminali

Super-calcolatori
Hanno molti processori, grandi memorie di massa e servono tipicamente centinaia o migliaia di terminali

2

RETI DI CALCOLATORI

- **Reti Locali:**
  connettono elaboratori ***fisicamente vicini***
  (nello stesso ufficio o stabilimento).
- **LAN (Local Area Network)**

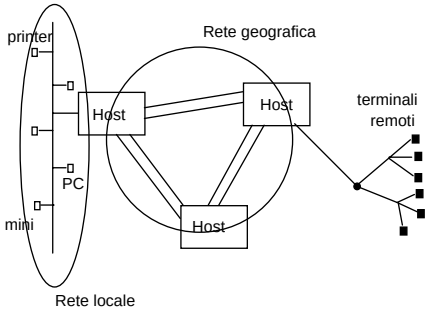


2

RETI DI CALCOLATORI (segue)

- **Reti geografiche (MAN, Metropolitan Area Network):**
  collegano elaboratori situati anche *a grande distanza (vari km)*.
- **WAN (Wide Area Network)**

  es: Internet



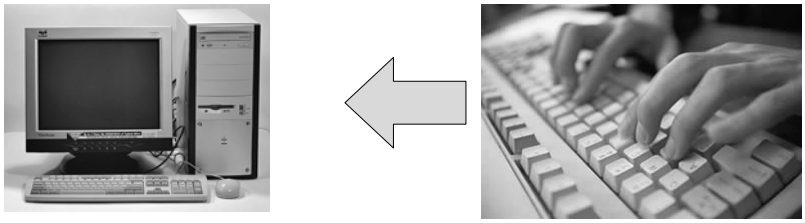
2

INTERNET: la rete delle reti

- **Internet:** la rete risultante dalla interconnessione mondiale di tutte le reti.
- Milioni di elaboratori ("**siti**") collegati **a ragnatela**
- **World-Wide Web (WWW)**

1

Linguaggi di Programmazione
dall'assembler ai linguaggi di alto livello



1

LINGUAGGIO MACCHINA

```
0   READ     8
1   READ     9
2   LOADA    8
3   LOADB    9
4   MUL
5   STOREA   8
6   WRITE    8
7   HALT
8   DATO INTERO
9   DATO INTERO
```

Rappresentazione reale (binaria):

0	0100	0000	0000	1000
1	0100	0000	0000	1001
2	0000	0000	0000	1000
3	0001	0000	0000	1001
4	1000	0000	0000	0000
5	0010	0000	0000	1000
6	0101	0000	0000	1000
7	1101	0000	0000	0000
8	0000	0000	0000	0000
9	0000	0000	0000	0000

1

ASSEMBLER

```
[0]  0100     8
[1]  0100     9
[2]  0000     8
[3]  0001     9
[4]  1000
[5]  0010     8
[6]  0101     8
[7]  1101
[8] (spazio disponibile)
[9] (spazio disponibile)
```

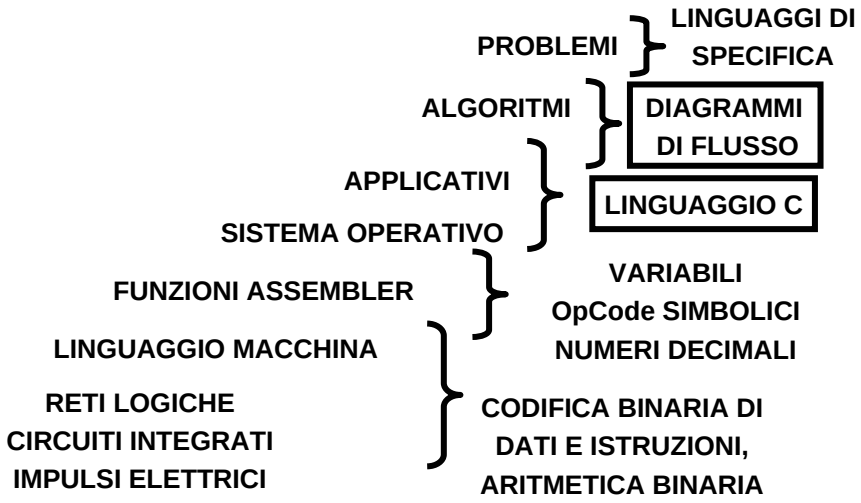
LINGUAGGIO MACCHINA

```
    READ     X
    READ     Y
    LOADA    X
    LOADB    Y
    MUL
    STOREA   X
    WRITE    X
    HALT
    X   INT
    Y   INT
```

ASSEMBLER

1

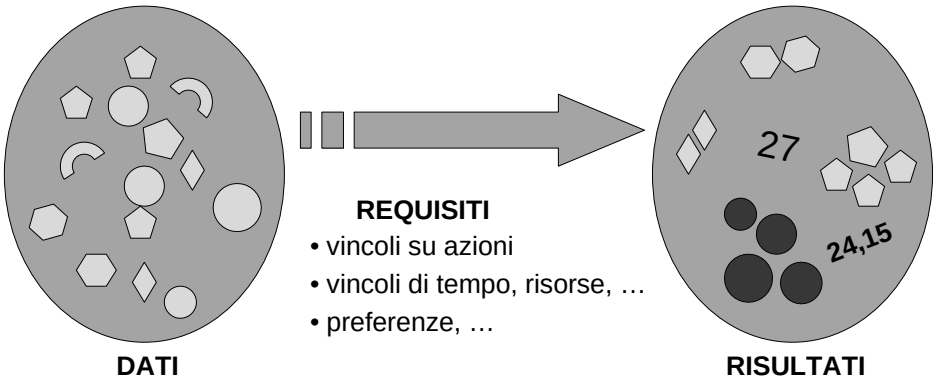
LIVELLI DI ASTRAZIONE



1

PROBLEMI E SPECIFICHE

→ Come specificare un problema?



REQUISITI
- vincoli su azioni
- vincoli di tempo, risorse, …
- preferenze, …

DATI
- tipi
- dominio
- modalità di acquisizione

RISULTATI
- tipi
- dominio
- modalità di restituzione

1

PROBLEMI E SPECIFICHE

→ Come specificare un problema?
(specifiche: dati, risultati, requisiti)

1. Dati *a* e *b*, risolvere l'equazione $ax + b = 0$
2. Stabilire se una parola viene alfabeticamente prima di un'altra
3. Somma di due numeri interi
4. Scrivere tutti gli n per cui l'equazione:
   $X^n + Y^n = Z^n$
   ha soluzioni intere (problema di Fermat)
5. Ordinare una lista di elementi
6. Calcolare il massimo comun divisore fra due numeri dati
7. Calcolare il massimo in un insieme

1

LINGUAGGI DI PROGRAMMAZIONE

- Un linguaggio di programmazione viene definito mediante:
  - **alfabeto** (o vocabolario)**:** stabilisce tutti i simboli che possono essere utilizzati nella scrittura di programmi
  - **sintassi:** specifica le regole grammaticali per la costruzione di frasi corrette (mediante la composizione di simboli)
  - **semantica:** associa un significato (ad esempio, in termini di azioni da eseguire) ad ogni frase sintatticamente corretta.

1

SEMANTICA

Attribuisce un significato ad ogni frase del linguaggio sintatticamente corretta.

- Molto spesso è definita **informalmente** (per esempio, a parole).
- Esistono **metodi formali** per definire la semantica di un linguaggio di programmazione in termini di…
  - azioni (semantica ***operazionale***)
  - funzioni matematiche (semantica ***denotazionale***)
  - formule logiche (semantica ***assiomatica***)

⇨ Benefici per
- ⇨ il **programmatore** (comprensione dei costrutti, prove formali di correttezza),
- ⇨ l'**implementatore** (costruzione del traduttore corretto),
- ⇨ il **progettista** di linguaggi (strumenti formali di progetto).

1

SINTASSI DI UN LINGUAGGIO DI PROGRAMMAZIONE

La sintassi di un linguaggio può essere descritta:
- in modo informale (ad esempio, *a parole*), oppure
- in modo formale (mediante una *grammatica formale*).

Grammatica formale:

è una notazione matematica che consente di esprimere in modo rigoroso la sintassi dei linguaggi di programmazione.

Un formalismo molto usato:

BNF (Backus-Naur Form)

1

GRAMMATICHE BNF

Una grammatica BNF è un insieme di 4 elementi:

1. un **alfabeto terminale** V: è l'insieme di tutti i simboli consentiti nella composizione di frasi sintatticamente corrette;
2. un **alfabeto non terminale** N (simboli indicati tra parentesi angolari **< … >**)
3. un insieme finito di **regole** (produzioni) P del tipo:

   X ::= A

   dove X$\in$ N, ed A è una sequenza di simboli α ("stringhe") tali che $\alpha \in (N \cup V)$.
4. un **assioma** (o simbolo iniziale, o scopo) S $\in$ N

1

ESEMPIO DI GRAMMATICA BNF

Esempio: il "linguaggio" per esprimere i naturali

V : { **0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9** }

N : { <naturale> , <cifra-non-nulla> , <cifra> }

P : <naturale> ::= **0** | <cifra-non-nulla> **{** <cifra> **}**
<cifra-non-nulla> ::= **1 | 2 | 3 | 4 | 5 | 6 | 7 | 9**
<cifra> ::= **0** | <cifra-non-nulla>

S : <naturale>

1

DERIVAZIONE

Una BNF definisce un **linguaggio** sull'alfabeto terminale, mediante un meccanismo di ***derivazione*** (o riscrittura):

Definizione:

Data una grammatica G e due stringhe β, γ elementi di $(N \cup V)^*$, si dice che

γ deriva direttamente da β

$(\beta \rightarrow \gamma)$

se le stringhe si possono decomporre in:

$\beta = \eta\ A\ \delta$ $\qquad$ $\gamma = \eta\ \alpha\ \delta$

ed esiste la produzione $A ::= \alpha$.

Si dice che **γ deriva da β** se: $\beta = \beta 0 \rightarrow \beta 1 \rightarrow \beta 2 \rightarrow ... \rightarrow \beta n = \gamma$

1

IL TOPO MANGIA IL GATTO

V : { **`il`** , **`gatto`** , **`topo`** , **`Tom`** , **`Jerry`** , **`mangia`** }

N : { <frase> , <soggetto> , <verbo> , <compl-oggetto> , <articolo> , <nome> }

S : <frase>

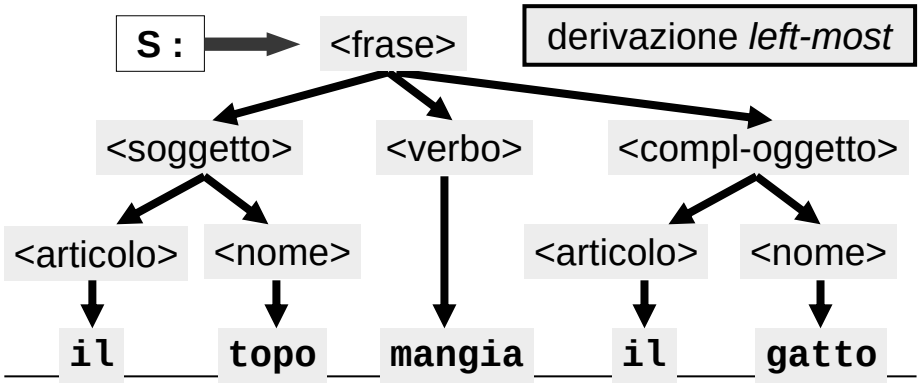
P (produzioni) :

<frase> ::= <soggetto> <verbo> <compl-oggetto>
<soggetto> ::= <articolo> <nome> | <nome-proprio>
<compl-oggetto> ::= <articolo> <nome> | <nome-proprio>
<articolo> ::= **`il`**
<nome> ::= **`gatto`** | **`topo`**
<nome-proprio> ::= **`Tom`** | **`Jerry`**
<verbo> ::= **`mangia`**

ALBERO SINTATTICO

Albero sintattico: esprime il processo di derivazione di una frase mediante una grammatica. Serve per analizzare la correttezza sintattica di una stringa di simboli terminali.

"**`il topo mangia il gatto`**"



LINGUAGGIO

Definizioni:

Data una grammatica G, si dice **linguaggio generato da G**, LG, l'insieme delle sequenze di simboli di V (frasi) derivabili, applicando le produzioni, a partire dal simbolo iniziale S.

Le frasi di un linguaggio di programmazione vengono dette **programmi** di tale linguaggio.

1

EBNF: BNF esteso

- **X ::= [a]B**
  a puo' comparire zero o una volta: equivale a **X ::= B | aB**
- **X ::= {a}^{n} B**
  a puo' comparire 0, 1, 2, …, n volte
  Es.: **X ::= {a}^{3} B** , equivale a **X ::= B | aB | aaB | aaaB**
- **X ::= {a} B**
  equivale a **X ::= B | aX** (*ricorsiva*)
  (a puo' comparire da 0 ad un massimo finito arbitrario di volte)
- **()** per raggruppare categorie sintattiche:
  Es.: **X ::= (a | b) D | c** equivale a **X ::= a D | b D | c**

1

ESEMPIO

Numeri interi di lunghezza qualsiasi con o senza segno
(non si permettono numeri con più di una cifra se quella più a sinistra è 0 es: 059)

V : {**0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9** } U { **+** , **-** }
N : { <intero> , <int-senza-segno> , <cifra> , <cifra-non-nulla> }
S : <intero>
P :

<intero>	::=	**[+ | -]** <intero-senza-segno>
<int-senza-segno>	::=	**0** | <cifra-non-nulla> **{** <cifra> **}**
<cifra>	::=	<cifra-non-nulla> | **0**
<cifra-non-nulla>	::=	**1 | 2 | 3 | 4 | 5 | 6 | 7 | 9**