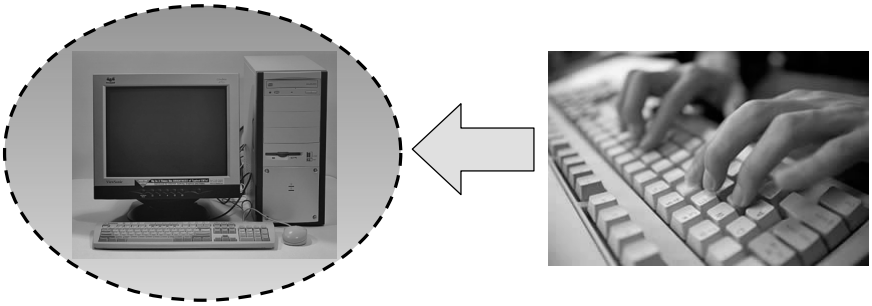
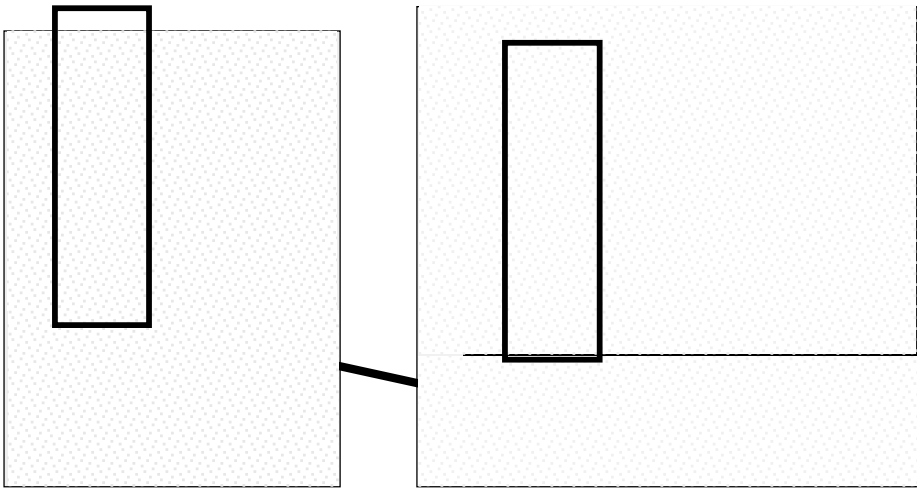
1

Linguaggi di Programmazione
dall'assembler ai linguaggi di alto livello



1

LINGUAGGIO MACCHINA



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

Rappresentazione reale (binaria):

0	READ	X	000
1	READ	Y	001
2	LOADA	X	000
3	LOADB	Y	001
4	MUL		000
5	STOREA	X	000
6	WRITE	X	000
7	HALT		000
8	X INT		000
9	Y INT		000

ASSEMBLER

1

LIVELLI DI ASTRAZIONE

PROBLEMI — LINGUAGGI DI SPECIFICA

ALGORITMI — DIAGRAMMI DI FLUSSO

APPLICATIVI — LINGUAGGIO C

SISTEMA OPERATIVO

FUNZIONI ASSEMBLER — VARIABILI, OpCode SIMBOLICI, NUMERI DECIMALI

LINGUAGGIO MACCHINA

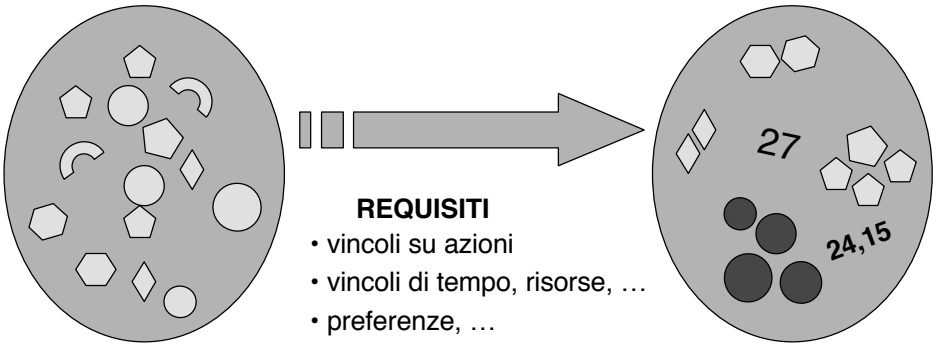
RETI LOGICHE
CIRCUITI INTEGRATI
IMPULSI ELETTRICI

CODIFICA BINARIA DI DATI E ISTRUZIONI, ARITMETICA BINARIA

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

Fondamenti di Informatica L- A

1

A…I

→ Come (far) riso…na in modo automatico?

(specifiche: pro…ioni elem.)

1. Dati *a* e *b*, riso… *ax + b = 0*
2. Stabilire se una… …beticamente 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

"J'ai trouvé une démonstration vraiment merveilleuse de cette propriété, mais la marge est trop étroite pour l'y écrire" (1637)

1

ALGORITMI EQUIVALENTI

Buoni o cattivi…

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:

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

ed esiste la produzione A ::= α.

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 *(**p**roduzioni) :*

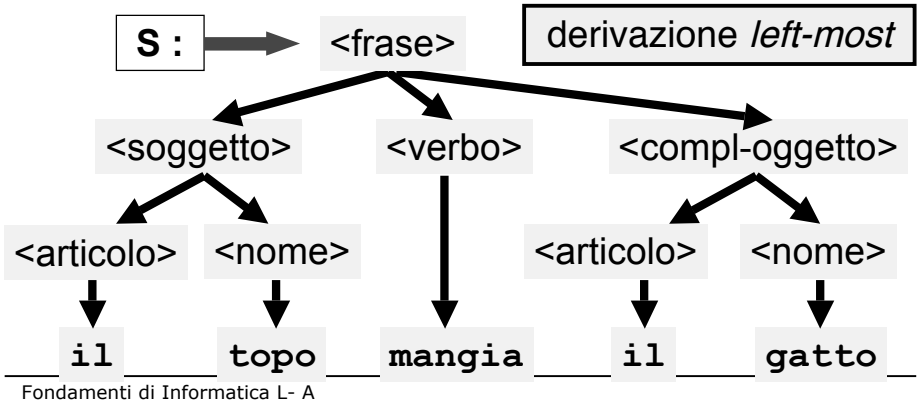
<frase> ::= <soggetto> <verbo> <compl-oggetto>
<soggetto> ::= <articolo> <nome> | <nome-proprio>
<compl-oggetto> ::= <articolo> <nome> | <nome-proprio>
<articolo> ::= **`il`**
<nome> ::= **`gatto`** | **`topo`**
<nome-proprio> ::= **`Tom`** | **`Jerry`**
<verbo> ::= **`mangia`**

1

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`**"



1

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**

ESEMPIO

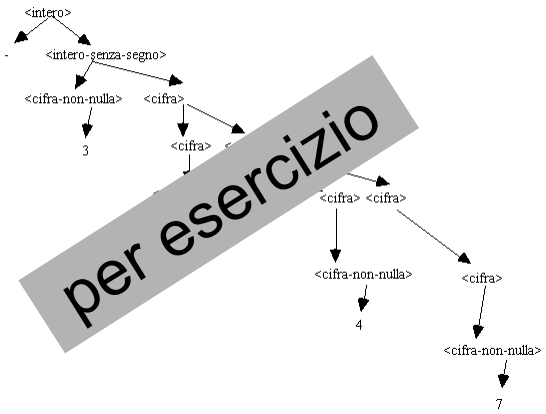
Numeri interi di lunghezza qualsiasi con o senza segno
(non si permettono numeri con più di una cifra se quella più a sinistra è 0 es: 059)

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**

Albero sintattico:

Albero sintattico per la generazione del numero ***-3547*** usando la grammatica EBNF:



per esercizio

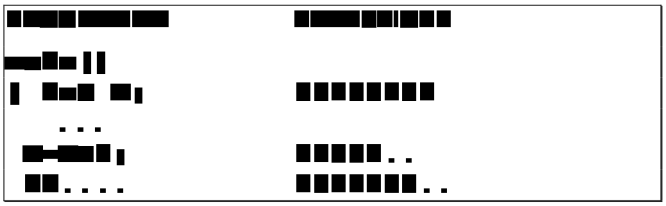
AMBIENTI DI PROGRAMMAZIONE

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***.

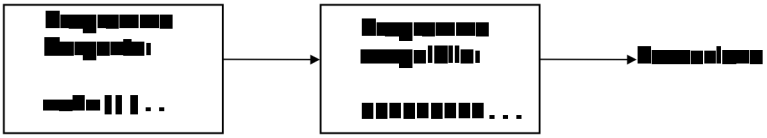
TRADUZIONE DI UN PROGRAMMA



Il traduttore converte

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

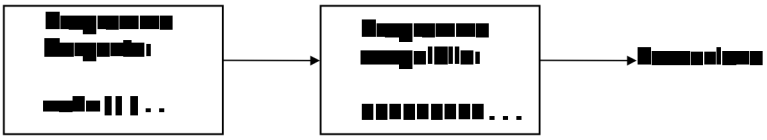
SVILUPPO DI PROGRAMMI



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.*

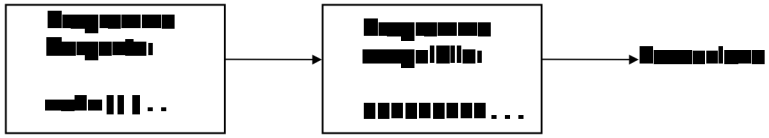
SVILUPPO DI PROGRAMMI



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.

SVILUPPO DI PROGRAMMI



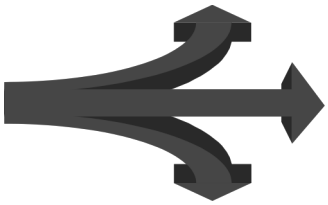
L'esecuzione di un programma ***compilato*** è **più veloce** dell'esecuzione di un programma ***interpretato***

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….



2

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.

Fondamenti di Informatica L- A

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.*