

I linguaggi di programmazione

AA2005/06 1 Programmazione
© Alberti 3. Linguaggi di programmazione

I linguaggi di programmazione

- I linguaggi di programmazione sono stati introdotti per facilitare ai programmatori il compito di scrittura dei programmi
- Sono linguaggi **simbolici**, in continua evoluzione
- Sono definiti da un insieme di regole formali, le regole grammaticali o **sintassi**
- Diversi **paradigmi di programmazione** per affrontare
 - il processo della programmazione
 - la traduzione dell'algoritmo nel linguaggio adottato

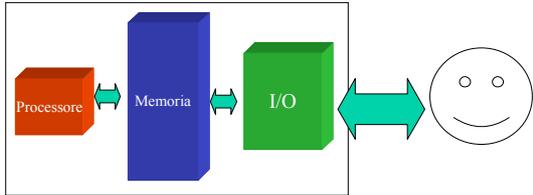
AA2005/06 2 Programmazione
© Alberti 3. Linguaggi di programmazione

La traduzione dei linguaggi

- Evoluzione verso sistemi di codici complessi e potenti, orientati più all'uomo che alla macchina
- La sfida degli anni '50 sulla traduzione dei linguaggi
- Linguaggi ad **alto livello** e a **basso livello** ovvero *i linguaggi macchina*

AA 2005/06 3 Programmazione
© Alberti 3. Linguaggi di programmazione

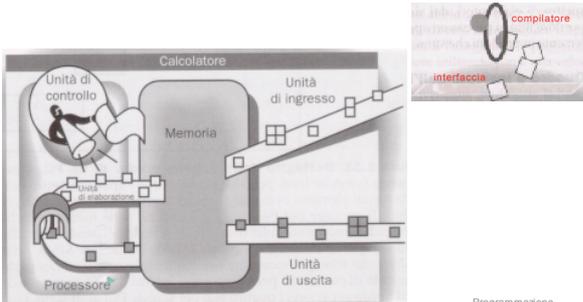
Schema dell'architettura



AA2005/06 4 Programmazione
© Alberti 3. Linguaggi di programmazione

Schema delle interconnessioni

- Le componenti sono tra loro interconnesse



AA 2005/06 3. Linguaggi di programmazione
© Alberti Programmazione

Il processore

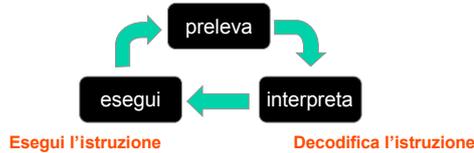
- **Il datapath o unità di elaborazione**
 - L'insieme dei circuiti che operano e manipolano i dati
- **Il controller**
 - L'insieme dei circuiti che interpretano un programma e sovrintendono all'esecuzione delle istruzioni da parte delle altre componenti del calcolatore

AA 2005/06 6 Programmazione
© Alberti 3. Linguaggi di programmazione

Ciclo del processore

- Il processore esegue in continuazione il ciclo
 - fetch-decode-execute*

Preleva dalla memoria principale la prossima istruzione di un programma



Il linguaggio del processore

- Ogni modello di microprocessore ha un *proprio linguaggio macchina* diverso da quello di altri processori
- Ogni modello di microprocessore *riconosce* solo programmi scritti nel proprio linguaggio macchina
- Il linguaggio macchina contiene *tutte e sole le operazioni* che possono essere eseguite dal microprocessore

Linguaggio macchina

- Esempio: un processore con 2 registri **R1** e **R2**
- Si debbono sommare i contenuti delle locazioni di memoria **var1** e **var2** e archiviare in **tot**
- L'operazione di somma è possibile solo sui dati archiviati nei registri
 - Trasferisci il contenuto di **var1** nel registro **R1**
 - Trasferisci il contenuto di **var2** nel registro **R2**
 - Somma il contenuto dei due registri
 - Trasferisci il risultato nella locazione di memoria **tot**

Linguaggio macchina – 2

- È necessario disporre di
- Istruzioni per il trasferimento di dati dalla memoria ai registri e viceversa
 - LOAD R x** **STORE R y**
 - Istruzioni aritmetiche/logiche
 - ADD R1 R2** **MUL R1 R2**
 - Eventualmente istruzioni di controllo e di salto
 - Salto incondizionato Salto condizionato
 - JUMP alfa** **JZERO R1 alfa**
 - ...
 - alfa: ...** **alfa: ...**

Esempio

- Sommare il contenuto di due variabili e salvarlo nella variabile **tot**

```
load R1, var1
load R2, var2
add R1, R2
store R1, tot
```

Diversi livelli di espressività

```
se a=b allora c:=0
altrimenti c:=a+b

load R1, a
load R2, b
sub R1, R2
jzero R1, fine

load R1, a
add R1, R2

fine: store R1, c
```

Algoritmo di Euclide in assembler

```
LOAD R1, 101
LOAD R2, 102
alfa: DIV R1, R2
MUL R1, R2
LOAD R2, 101
SUB R2, R1
JZERO R2, fine
LOAD R1, 102
STORE R1, 101
STORE R2, 102
JUMP alfa
fine: LOAD R1, 102
STORE R1, 103
```

AA 2005/06
© Alberti

Programmazione
3. Linguaggi di programmazione

Linguaggi di basso livello

- In un linguaggio macchina, ogni istruzione è una sequenza di cifre binarie
 - Totalmente illeggibile per l'uomo
 - Perfettamente non ambiguo per la macchina

```
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
```

traduzione delle istruzioni: LOAD x
 ADD y
 STORE z

AA 2005/06
© Alberti

14

Programmazione
3. Linguaggi di programmazione

Le operazioni elementari

- Ogni istruzione del linguaggio macchina viene eseguita da un microprocessore svolgendo una serie di passi: le *operazioni elementari*
- Il numero di operazioni elementari necessario a portare a compimento un'istruzione in linguaggio macchina è dell'ordine di 7-10

AA 2005/06
© Alberti

15

Programmazione
3. Linguaggi di programmazione

Linguaggi macchina: problemi

- Sono specifici della macchina
- Occorre conoscere l'architettura della macchina per scrivere programmi
- I programmi non sono trasportabili
- I codici sono poco leggibili
- I programmatori si specializzano nel cercare efficienza su una macchina specifica, anziché concentrarsi sul problema

AA 2005/06
© Alberti

16

Programmazione
3. Linguaggi di programmazione

Traduzione dei linguaggi

- Il *concetto di traduzione* dei linguaggi ha permesso l'evoluzione verso sistemi simbolici più espressivi e più facilmente manipolabili dai programmatori
- Il programmatore scrive un programma in un linguaggio ad alto livello senza preoccuparsi della macchina che esegue il programma
- Il *compilatore* viene predisposto per una piattaforma specifica e poi traduce tutti i programmi scritti in uno specifico linguaggio ad alto livello

AA 2005/06
© Alberti

17

Programmazione
3. Linguaggi di programmazione

Programma compilatore

- Un compilatore di un linguaggio L (Pascal, C, FORTRAN, ...) per una macchina X, è un programma che:
 - preso in ingresso P, il programma *sorgente*, scritto nel linguaggio L
 - produce in uscita P', il programma *oggetto*, equivalente a P scritto nel linguaggio della macchina X

AA 2005/06
© Alberti

18

Programmazione
3. Linguaggi di programmazione

Programma compilatore

- Il programma **sorgente** è indipendente dalla piattaforma
- Il programma **oggetto** è dipendente dalla piattaforma
- Il compilatore può tradurre le istruzioni scritte in un linguaggio ad alto livello, poiché questo è descritto da regole sintattiche precise
 - Necessità di strumenti formali per la descrizione delle regole (**tavole sintattiche, grammatiche, BNF**)

AA 2005/06
© Alberti

19

Programmazione
3. Linguaggi di programmazione

Esigenza di una macchina astratta

- Il significato delle istruzioni in linguaggio macchina è descritto in termini di funzionamento del processore (**semantica operativa**)
 - **LOAD R1 102**
 - ha l'effetto di copiare il valore della locazione di memoria **102** nel registro **R1**

....

AA 2005/06
© Alberti

20

Programmazione
3. Linguaggi di programmazione

Il ruolo della macchina astratta

- I linguaggi ad alto livello introducono un **livello d'astrazione** rispetto all'architettura della macchina
- Il significato delle istruzioni di un linguaggio ad alto livello è descritto riferendosi ad una macchina astratta
- Il programmatore pensa in termini delle operazioni della macchina astratta e non del processore

AA 2005/06
© Alberti

21

Programmazione
3. Linguaggi di programmazione

Programma interprete

- Un interprete è un programma che simula direttamente una macchina astratta
 - Legge ogni istruzione contenuta nel programma P, ed effettua immediatamente, utilizzando la macchina sottostante, le operazioni corrispondenti al suo significato
- I linguaggi compilati
 - Pascal, C
- I linguaggi interpretati
 - Lisp, Prolog

AA 2005/06
© Alberti

22

Programmazione
3. Linguaggi di programmazione

Sintassi e semantica

- Le **regole di sintassi** definiscono come si devono comporre i simboli e le parole per formare istruzioni corrette
- La **semantica** di un'istruzione definisce cosa questa significhi (lo scopo dell'istruzione)
- Un programma sintatticamente corretto non è necessariamente semanticamente corretto
- I programmi fanno quello che prescriviamo che facciano e non quello che vorremmo che facessero

AA 2005/06
© Alberti

23

Programmazione
3. Linguaggi di programmazione

Sintassi

- Il sistema di regole formali che definisce il linguaggio
 - Le parole, i simboli e il modo di organizzarli
- Consente di stabilire se un'istruzione è **ben formata**
 - È corretto scrivere **se $x + n > m$ allora STOP** ?
- Facilitano il compito al programmatore
- I programmi devono essere tradotti nel linguaggio nativo della macchina

AA 2005/06
© Alberti

24

Programmazione
3. Linguaggi di programmazione

Descrivere le regole sintattiche

- Formalismi per scrivere le regole sintattiche dei linguaggi:
 - Grammatiche**
 - Forma di Backus-Naur (BNF)**
 - Tavole sintattiche** o grafi sintattici

AA 2005/06
© Alberti

25

Programmazione
3. Linguaggi di programmazione

BNF

La sintassi dei numeri reali

```
<reale> ::= <sequenza_cifre> .
           <sequenza_cifre>
<sequenza_cifre> ::= <cifra> | <cifra>
                    <sequenza_cifre>
<cifra> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

AA 2005/06
© Alberti

26

Programmazione
3. Linguaggi di programmazione

Grammatica

- Una grammatica G è una quaterna (T, N, P, S)
- T è un insieme finito di **simboli terminali**
- un insieme finito N di **simboli non terminali**: i *metasimboli*
- un insieme finito P di **regole di produzione**
- un **simbolo iniziale** S, appartenente all'insieme dei simboli non terminali, utilizzato come punto di partenza nella costruzione delle sentenze

AA 2005/06
© Alberti

27

Programmazione
3. Linguaggi di programmazione

Primo esempio

T = {**Paolo, Francesca, legge, dorme**}

N = {*frase, soggetto, predicato*}

S = *frase*

P l'insieme delle regole espresse in *BNF (Backus- Naur form)*:

```
frase ::= soggetto predicato
soggetto ::= Paolo | Francesca
predicato ::= legge | dorme
```

AA 2005/06
© Alberti

28

Programmazione
3. Linguaggi di programmazione

Linguaggio generato

- Una grammatica consente di **produrre** frasi del linguaggio e di **decidere** quali frasi vi appartengono
- Il linguaggio generato da G è l'insieme di tutte le sequenze di simboli terminali ottenibili applicando le regole di produzione dell'insieme P, a partire dal simbolo iniziale S
- L'esempio:
 - Paolo legge, Francesca dorme**
 - Ma non legge Paolo, Dario dorme**

AA 2005/06
© Alberti

29

Programmazione
3. Linguaggi di programmazione

Un altro esempio

T = {**il, lo, la, cane, mela, gatto, mangia, graffia, ,**}

N = {*frase, soggetto, verbo, complemento, articolo, nome*}

S = *frase*

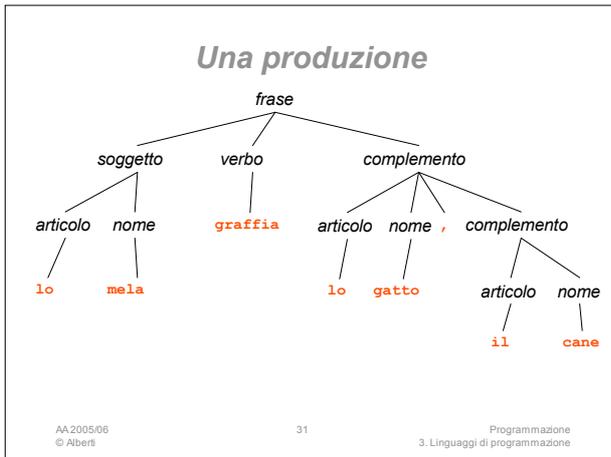
P l'insieme delle regole espresse in *BNF (Backus- Naur form)*:

```
frase ::= soggetto verbo complemento
soggetto ::= articolo nome
articolo ::= il | la | lo
nome ::= cane | mela | gatto
verbo ::= mangia | graffia
complemento ::= articolo nome | articolo nome,
```

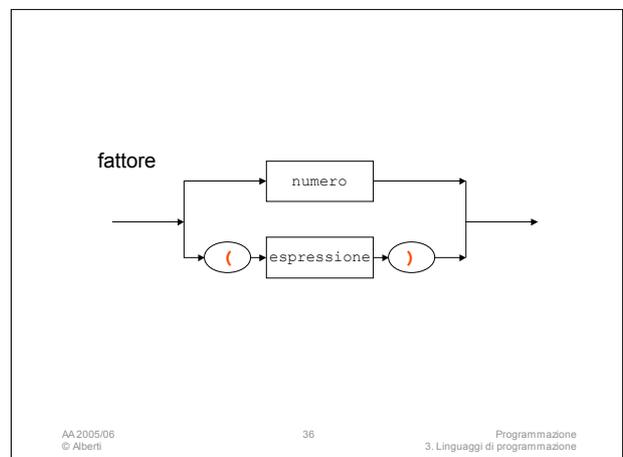
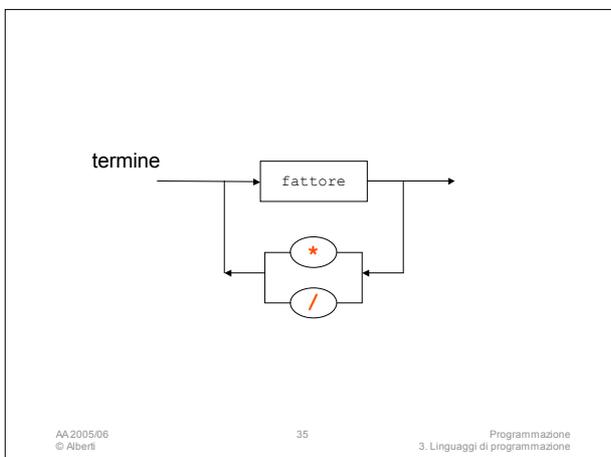
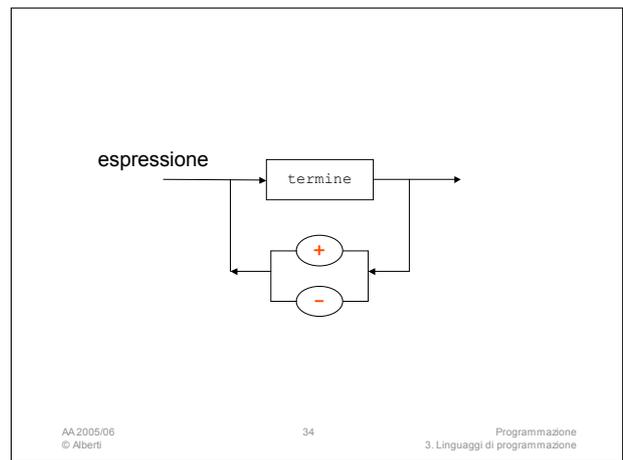
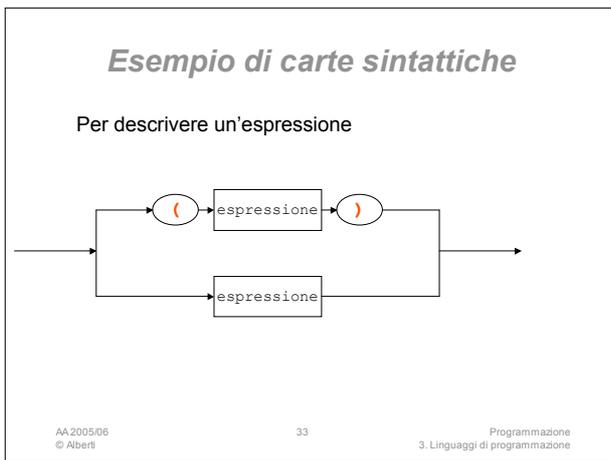
AA 2005/06
© Alberti

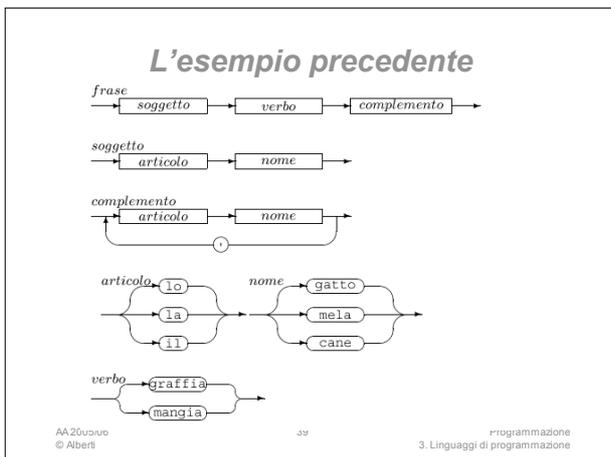
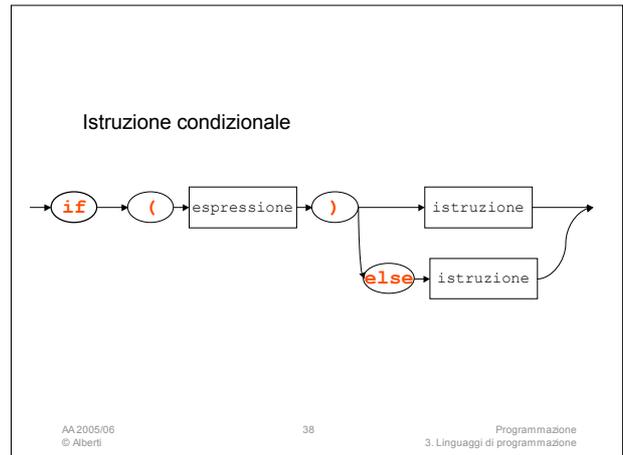
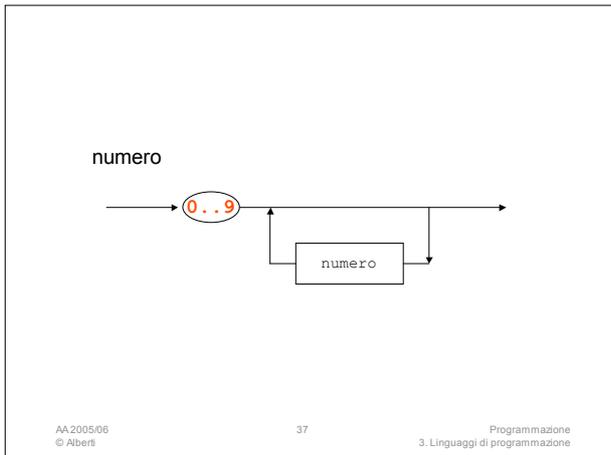
30

Programmazione
3. Linguaggi di programmazione



- ### Le carte sintattiche
- Le regole di produzione possono essere espresse per mezzo di diagrammi detti carte sintattiche
 - Una carta sintattica per ciascun simbolo non terminale della grammatica
 - i rettangoli indicano simboli non terminali (che andranno espansi con le carte sintattiche corrispondenti)
 - gli ovali indicano simboli terminali, che quindi non devono essere espansi ulteriormente
 - ogni biforcazione indica un'alternativa
- AA 2005/06 © Alberti 32 Programmazione 3. Linguaggi di programmazione





Semantica

- Specifica il significato di un programma
- Esempio: una sintassi per descrivere le date:


```
<data> ::=
            <cfr><cfr> . <cfr><cfr> . <cfr><cfr><cfr><cfr>
            <cfr> ::= 0|1|2|3|4|5|6|7|8|9
```
- Quindi 01.02.2001 è una data
- Ma il giorno a cui questa data si riferisce non è identificato dalla sintassi
 - 01.02.2001 in USA identifica il 2 Gennaio 2001
 - 01.02.2001 in Europa identifica il 1 Febbraio 2001
- La stessa frase ha significati diversi in contesti diversi

AA 2005/06 © Alberti 40 Programmazione 3. Linguaggi di programmazione