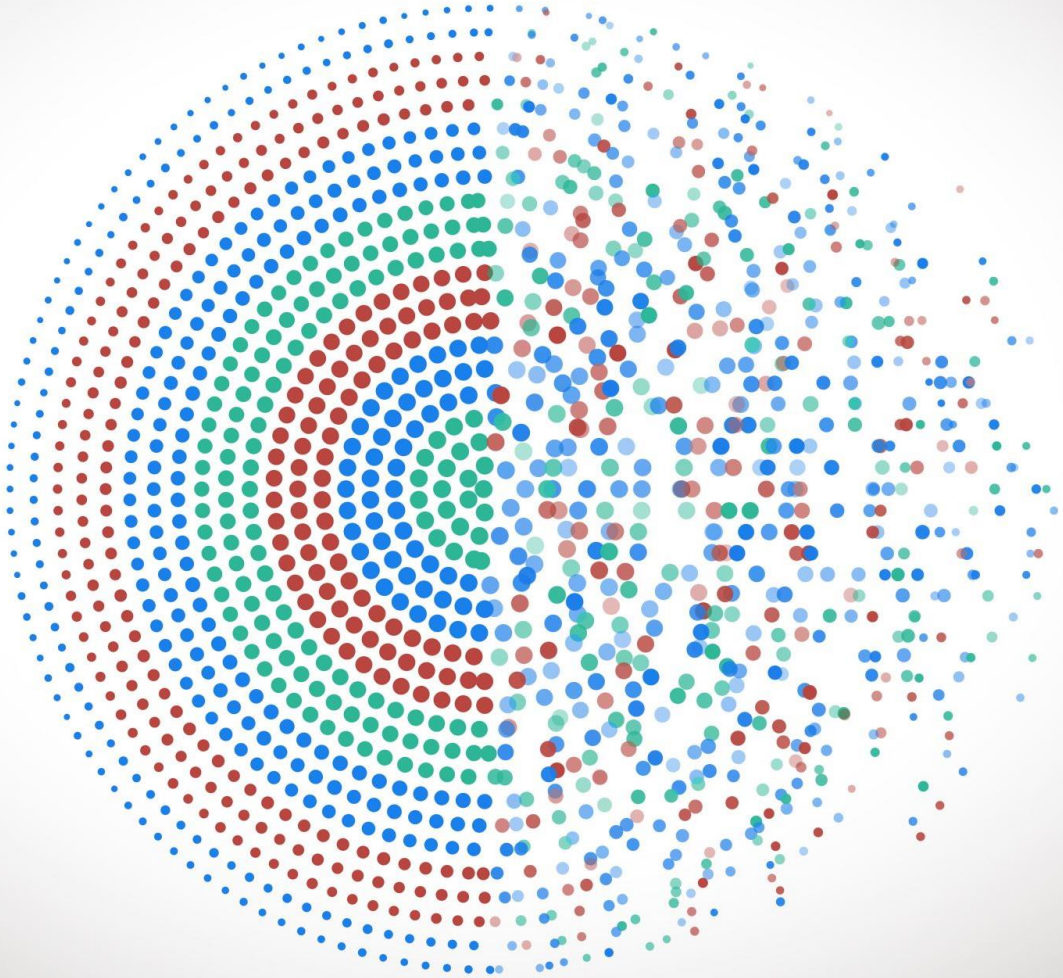
# Fondamenti di programmazione

a.a. 2022-23

Antinisca Di Marco
antinisca.dimarco@univaq.it



# Modulo Integrato

Fondamenti di Programmazione con Laboratorio (12 CFU). Due moduli:

- **Modulo di Fondamenti di Programmazione** (6 CFU)
  - Antinisca DI MARCO
- Modulo di Laboratorio di Programmazione I (6 CFU)
  - Monica NESI

# Fondamenti di Programmazione (?)

- I fondamenti di programmazione sono quelle **conoscenze basilari** che occorre possedere per la **programmazione** informatica **in un qualunque linguaggio di programmazione**

# Testi di riferimento

Modulo di Fondamenti di Programmazione (6 CFU)

- **Elementi di Sintassi dei Linguaggi di Programmazione** - Corso di Laurea in Informatica Università di Pisa a.a. 2004/05, R. Barbuti, P. Mancarella, D. Pedreschi, F. Turini (*)
- **Elementi di Semantica Operazionale**, R. Barbuti, P. Mancarella e F. Turini (*) Questa dispensa rivisita la dispensa precedente "Semantica Operazionale". Gli studenti posso scegliere una delle due.
- **Semantica Operazionale di +/- Java** (versione del 03 Dicembre 2010), M. Autili, P. Inverardi.

Le precedenti dispense sono reperibile sul TEAM del corso, canale generale: File-> materiale del corso

(*) Le prime due dispense sono comunque disponibili anche on-line nel sito ufficiale dell'Università di Pisa. Sono state pubblicate anche nelle precedenti pagine solo per convenienza degli studenti.

# Parti da Studiare

- Per la dispensa "**Elementi di Sintassi dei Linguaggi di Programmazione**" il corso copre la parte riguardante le grammatiche ma non copre la parte riguardante gli automi. Le parti da studiare sono:
  - da pag. 1 a pag. 5
  - dalla Sezione 3 Grammatiche (cioè da pag. 19) fino alla Sezione 3.4 compresa (cioè fino pag. 40)
- Per la dispensa "**Elementi di Semantica Operazionale**" la parte da studiare va:
  - dall'inizio fino alla Sezione 5.4 compresa (cioè fino pag. 48).
- La dispensa "**Semantica Operazionale +/-Java**" va studiata tutta

# Modalità d’esame

- Due prove parziali scritte sufficienti oppure Prova totale scritta
- Orale obbligatorio dopo il secondo parziale o dopo la prova totale

- La prima prova parziale è a metà corso e prevede gli argomenti delle prime due dispense

# Cosa sono i fondamenti di programmazione?

# Cos’è un linguaggio di programmazione

- Un linguaggio di programmazione è una notazione formale che può essere usata per descrivere algoritmi.

- Due aspetti del linguaggio:
  - sintassi
  - semantica

# Sintassi e Semantica

- **Sintassi**: l'insieme di regole formali per la scrittura di programmi in un linguaggio, che dettano le modalità per costruire frasi corrette nel linguaggio stesso
- **Semantica**: l'insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

# Esempi in Italiano

- Struttura di una frase semplice (regola sintattica):

<soggetto> <verbo> <complemento-oggetto>

- Esempi di frasi **corrette** per quanto riguarda la **struttura sintattica**:
  - Il bimbo mangia il gelato
  - Il papa stira la camicia
  - La ghianda prepara lo zaino
  - Il cane rincorre la palla

**NB: una frase può essere sintatticamente corretta e tuttavia non avere significato!**

# Linguaggio

- È un insieme di frasi (sintatticamente) ammissibili.

Dato un vocabolario, o **alfabeto**, di elementi di base (simboli o **elementi terminali**), un **linguaggio è un sottoinsieme di tutte le** frasi (o **stringhe**) ottenibili come sequenze **di elementi terminali**.

# Esempio

- Consideriamo il linguaggio delle espressioni aritmetiche ottenibili a partire da numeri interi e dalle quattro operazioni +, −, × e /.
- Fra tutte le possibili sequenze (o stringhe) di numeri ed operazioni aritmetiche:
  - ve ne sono di ammissibili, come 3 × 4 + 2,
  - e di non ammissibili, come 3 + × + 4.
- Il linguaggio delle espressioni aritmetiche può essere identificato con il sottoinsieme delle stringhe ammissibili.
- Un linguaggio di programmazione può essere identificato con l'insieme delle proprie stringhe ammissibili, che comunemente chiamiamo **programmi**.

# Esempio

- Consideriamo il linguaggio delle espressioni aritmetiche ottenibili a partire da numeri interi e dalle quattro operazioni +, −, × e /.
- Fra tutte le pos aritmetiche:
  - ve ne sono
  - e di non am
- Il linguaggio d con il sottoinsieme d
- Un linguaggio di programmazione può essere identificato con l'insieme delle proprie stringhe ammissibili, che comunemente chiamiamo **programmi**.

**Descrivere un linguaggio significa descrivere l'insieme delle stringhe del linguaggio stesso**

# Formalizzazione del concetto di linguaggio

Concetti preliminari:

- Alfabeto: insieme finito di simboli detti terminali, lo indichiamo con $\Lambda$
- Stringa sull'alfabeto $\Lambda$ : una stringa `e una sequenza finita di simboli di $\Lambda$ $a_1 a_2 \cdot \cdot \cdot a_n$, con $n \geq 0$, dove ciascun $a_i$ elemento di $\Lambda$.

  In formule, $\forall i \in [1, n].a_i \in \Lambda$.

Se l'alfabeto $\Lambda$ è l'insieme {0, 1}, un esempio di stringa è 0011011000l101.

# Formalizzazione del concetto di linguaggio

Concetti preliminari:

- Consideriamo una stringa **s** = $a_1 a_2 \cdots a_n$ con $n \geq 0$
- Il numero naturale **n** è detto lunghezza della stringa.
- Se la lunghezza di una stringa s è 0, allora la stringa s è chiamata stringa vuota, ed è rappresentata con il simbolo **ε**.
- L'insieme di tutte le stringhe su un dato alfabeto **Λ** è indicato come **Λ∗**.
- In generale, il simbolo $*$ è un operatore che, dato un insieme S, costruisce l'insieme di tutte le stringhe sull'insieme S.

# Formalizzazione del concetto di linguaggio

Concetti preliminari:

- Si noti che, se $\Lambda$ non è vuoto, allora $\Lambda^*$ è un insieme infinito, formato cioè da infiniti elementi, anche se ciascuno di questi è di lunghezza finita.

**Un linguaggio su un alfabeto $\Lambda$ è un sottoinsieme di $\Lambda^*$, ovvero un insieme di stringhe sull'alfabeto.**

**In formule, L è un linguaggio su $\Lambda$ se $L \subseteq \Lambda^*$.**

# Formalizzazione del concetto di linguaggio

- $\Lambda^*$ stesso e l'insieme vuoto $\emptyset$ sono due esempi di linguaggi, entrambi poco interessanti.
- I linguaggi "interessanti", su un dato alfabeto , saranno infatti insiemi non vuoti e non banali di stringhe che hanno in comune una qualche proprietà rilevante.
- Sui linguaggi, visti come insiemi di stringhe, sono definite tutte le operazioni insiemistiche: unione ($\cup$), intersezione ($\cap$), complemento e quelle da esse derivate.
- E' definita inoltre l'operazione di concatenazione, basata naturalmente sulla concatenazione di stringhe: siano $L1, L2 \subseteq \Lambda^*$

  $L1L2 = \{\alpha\beta \mid \alpha \in L1 \wedge \beta \in L2\}$

Fine

# Grammatiche

- Le grammatiche libere da contesto sono un strumento **generativo** (**ricorsivo**) per descrivere linguaggi
- le grammatiche costituiscono una notazione concisa per descrivere la sintassi di un linguaggio di programmazione.

# Grammatiche

- Le espressioni aritmetiche possono essere definite ricorsivamente in modo naturale. Consideriamo le espressioni aritmetiche che contengono:
  - i quattro operatori binari +, −, ∗ e /;
  - le parentesi;
  - i numeri come operandi.

Tali espressioni vengono di solito definite in modo induttivo come segue:

**Base**. Un numero è un'espressione.

**Induzione**. Se E è un'espressione, lo sono anche:

a) (E)

b) E + E

c) E − E

d) E ∗ E;

e) E/E.

# Grammatiche

- Le espressioni aritmetiche possono essere definite ricorsivamente in modo naturale. Consideriamo le espressioni aritmetiche che contengono:
  - i quattro operatori binari +, −, ∗ e /;
  - le parentesi;
  - i numeri come operandi.

Tali espressioni vengono di solito definite in mo

**Base**. Un numero è un'espressione.

**Induzione**. Se E è un'espressione, lo sono anch

a) (E)

b) E + E

c) E − E

d) E ∗ E;

e) E/E.

**Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica**

**<Espressione> → <Numero>**
**<Espressione> → ( <Espressione> )**
**<Espressione> → <Espressione> + <Espressione>**
**<Espressione> → <Espressione> - <Espressione>**
**<Espressione> → <Espressione> * <Espressione>**
**<Espressione> → <Espressione> / <Espressione>**

# Grammatiche

- <Espressione> viene detto categoria sintattica e sta per una qualunque stringa nel linguaggio delle espressioni aritmetiche.
- Il simbolo → significa "può essere composto da": per esempio, la regola <Espressione> → ( <Espressione> ) dice che un'espressione può essere composta da una parentesi aperta, seguita da una qualunque stringa che sia un'espressione, seguita da una parentesi chiusa.

**Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica**

**<Espressione> → <Numero>**
**<Espressione> → ( <Espressione> )**
**<Espressione> → <Espressione> + <Espressione>**
**<Espressione> → <Espressione> - <Espressione>**
**<Espressione> → <Espressione> * <Espressione>**
**<Espressione> → <Espressione> / <Espressione>**

# Grammatiche

- Nella prima regola il simbolo <Numero> a destra della freccia è anch'esso una categoria sintattica, da interpretarsi come un segnaposto per una generica stringa che possa essere interpretata come un numero.
- Allo stato attuale non ci sono regole in cui <Numero> compaia a sinistra della freccia, e quindi non è ancora definito quali stringhe possano essere usate per rappresentare i numeri.

**Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica**

**<Espressione> → <Numero>**
**<Espressione> → ( <Espressione> )**
**<Espressione> → <Espressione> + <Espressione>**
**<Espressione> → <Espressione> - <Espressione>**
**<Espressione> → <Espressione> * <Espressione>**
**<Espressione> → <Espressione> / <Espressione>**

# Grammatiche

- Una grammatica è costituita da una o più produzioni : ogni linea della riquadro affianco è una produzione.
- In generale, una produzione `e formata da tre parti:

1. una testa, che è la categoria sintattica a sinistra della freccia;
2. il metasimbolo →;
3. il corpo, costituito da 0 o più categorie sintattiche e/o simboli terminali a destra della freccia.

**Le grammatiche consentono di scrivere queste regole in modo conciso e con un significato preciso. Come esempio, la nostra definizione delle espressioni aritmetiche potrebbe essere data mediante la grammatica**

**<Espressione> → <Numero>**
**<Espressione> → ( <Espressione> )**
**<Espressione> → <Espressione> + <Espressione>**
**<Espressione> → <Espressione> - <Espressione>**
**<Espressione> → <Espressione> * <Espressione>**
**<Espressione> → <Espressione> / <Espressione>**

# Convenzioni

- <S> → ε significa che la stringa vuota fa parte del linguaggio della categoria sintattica <S>.
- <S> → B1, <S> → B2, . . . ,<S> → Bn  è equivalente a

<S> → B1 | B2 | · · · | Bn

# Completiamo la grammatica delle espressioni

<Espressione> → <Numero>
<Espressione> → ( <Espressione> )
<Espressione> → <Espressione> + <Espressione>
<Espressione> → <Espressione> - <Espressione>
<Espressione> → <Espressione> * <Espressione>
<Espressione> → <Espressione> / <Espressione>
<Numero> → ????

# Grammatica

- Una **grammatica** G è definita come una **quadrupla**

# Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

$\Lambda$ è un insieme di simboli detto alfabeto

# Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

$\Lambda$ è un insieme di simboli detto alfabeto

$V$ è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

# Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

$\Lambda$ è un insieme di simboli detto alfabeto

$V$ è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

$S \in V$ è la categoria sintattica principale o iniziale

# Grammatica

- Una **grammatica** G è definita come una **quadrupla**

$$< \Lambda, V, S, P >$$

$\Lambda$ è un insieme di simboli detto alfabeto

$V$ è l'insieme, finito, delle categorie sintattiche, ovvero di variabili che rappresentano sotto-linguaggi

$S \in V$ è la categoria sintattica principale o iniziale

$P$ è un insieme finito di produzioni. Ciascuna produzione, nel caso delle grammatiche libere ha la struttura $A \rightarrow \alpha$

$A \in V$
$\alpha \in (\Lambda \cup V)^+$

## Esercizi: Definire una grammatica G per ognuno dei seguenti linguaggi

$L(G1) = \{ a^n \mid n \geq 1 \}$

$L(G2) = \{ a^n \mid n \geq 0 \}$

$L(G3) = \{ a^n b^m \mid n \geq 1, m \geq 0 \}$

$L(G4) = \{ a^n b^n \mid n \geq 1 \}$

$L(G5) = \{ a^n (bc)^m \mid n \geq 0, m \geq 0 \}$

$L(G6) = \{ a^n b^m c^n \mid n \geq 0, m \geq 0 \}$