



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fondamenti di Informatica T

Introduzione all'informatica

Iniziamo dalle Basi

Che cos'è l'informatica?

Non è facile da definire!

Alcune affermazioni vere:

- L'informatica è parente stretta della matematica
- Ha a che fare con il modo in cui risolviamo i problemi
- Si può fare anche senza un computer

Vediamo qualche esempio...



Un Esempio

Determinare il massimo in un insieme di numeri



Un Esempio

Un possibile procedimento risolutore:

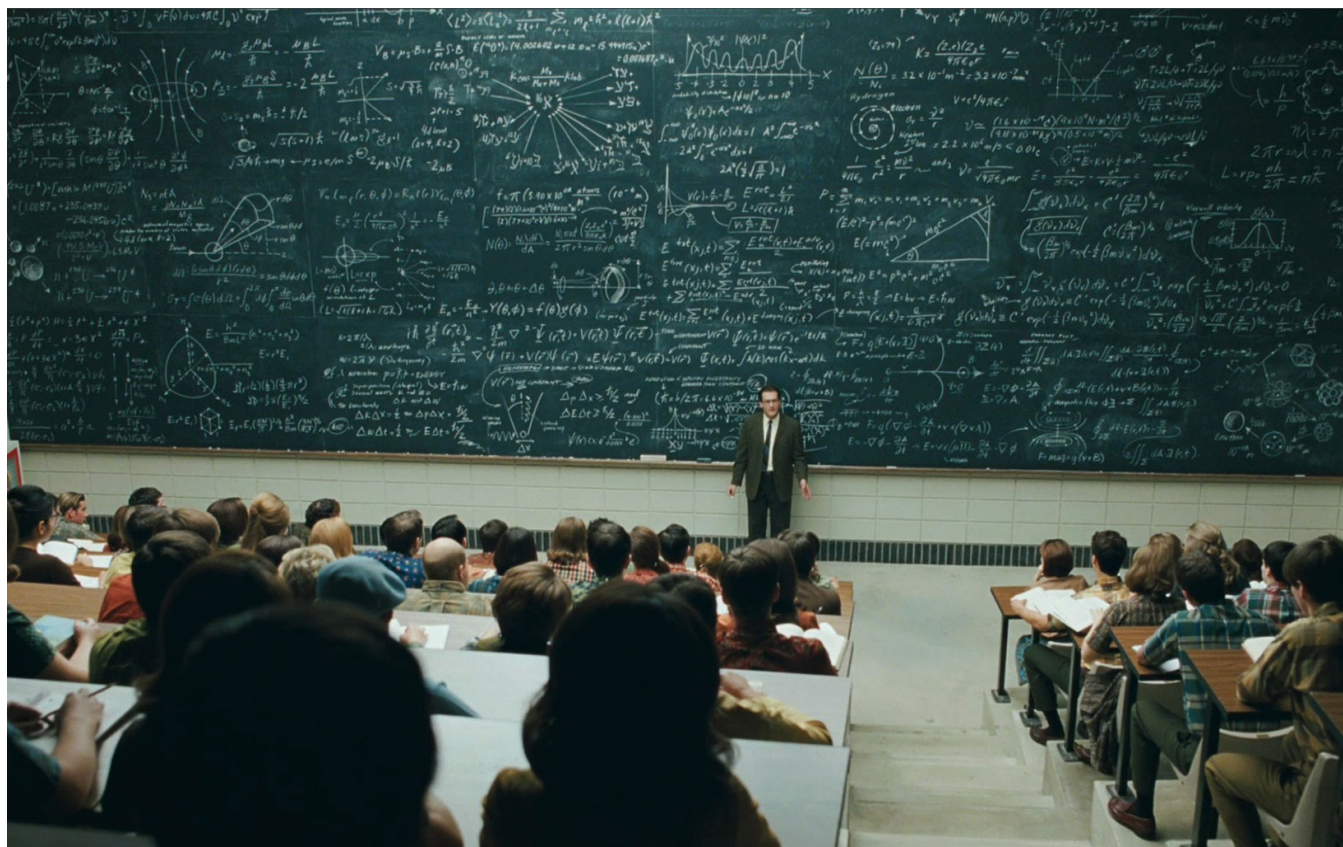
- Sia x il valore della prima tessera
- Per ogni altra tessera y :
 - Se $y > x$, allora $x = y$

Alla fine del procedimento x contiene il massimo



Un Esempio

Riposizionarci in ordine alfabetico:



Un Esempio

Un possibile procedimento risolutore:

- Consideriamo tutte le coppie di studenti adiacenti
- Se la coppia non è in ordine, si scambia
- Ripetere finché le coppie non sono in ordine

Alla fine del procedimento l'aula è ordinata



Un Esempio

Sommare due numeri X e Y:



...Usando come base degli incrementi semplici (+1)



Un Esempio

Un possibile procedimento risolutore:

- $Z = X$
- Ripeti Y volte:
 - $Z = Z + 1$

Alla fine del procedimento Z vale $X + Y$



Algoritmi

Quelli che abbiamo visto sono esempi di **algoritmi**

Informalmente:

*Un algoritmo è **un processo che risolve un problema***

Un filo più nel dettaglio:

- Specifica una sequenza di operazioni elementari...
- ...Che se eseguite, producono la soluzione del problema

Riprenderemo questi concetti in profondità più avanti



Algoritmi ed Elaboratori

Una considerazione importante

- Chi decide quali operazioni elementari siano disponibili?
 - E.g. “solo incrementi semplici”
- Chi decide cosa facciano?
 - E.g. incremento $\rightarrow X = X + 1$

Per eseguire un algoritmo è necessario un elaboratore

È l'elaboratore che determina:

- Quali **tipi di dato** possiamo manipolare
- Quali **operazioni** siano disponibili e la loro **semantica**



Algoritmi ed Elaboratori

Una considerazione importante

- Chi decide quali operazioni elementari siano disponibili?
 - E.g. “solo incrementi semplici”
- Chi decide cosa facciano?
 - E.g. incremento $\rightarrow X = X + 1$

Per eseguire un algoritmo è necessario un elaboratore

Un **elaboratore** è una entità che può:

- **Memorizzare** informazioni
- Eseguire su di esse alcune **operazioni elementari**



Algoritmi ed Elaboratori

Questo è un elaboratore:



Algoritmi ed Elaboratori

Questo è un elaboratore:



Algoritmi ed Elaboratori

Questo è un elaboratore:



Algoritmi ed Elaboratori

Questo è un elaboratore:



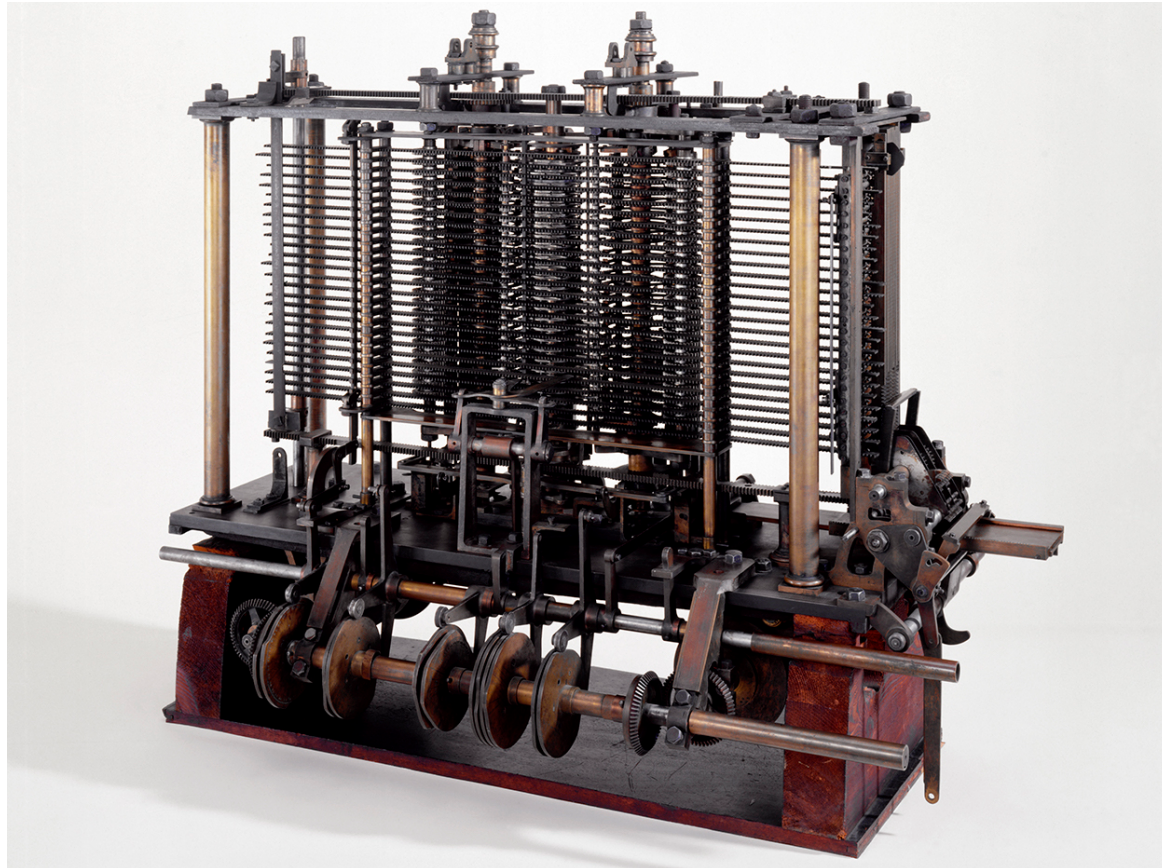
Algoritmi ed Elaboratori

Questo è un elaboratore (Apple I, 1976):



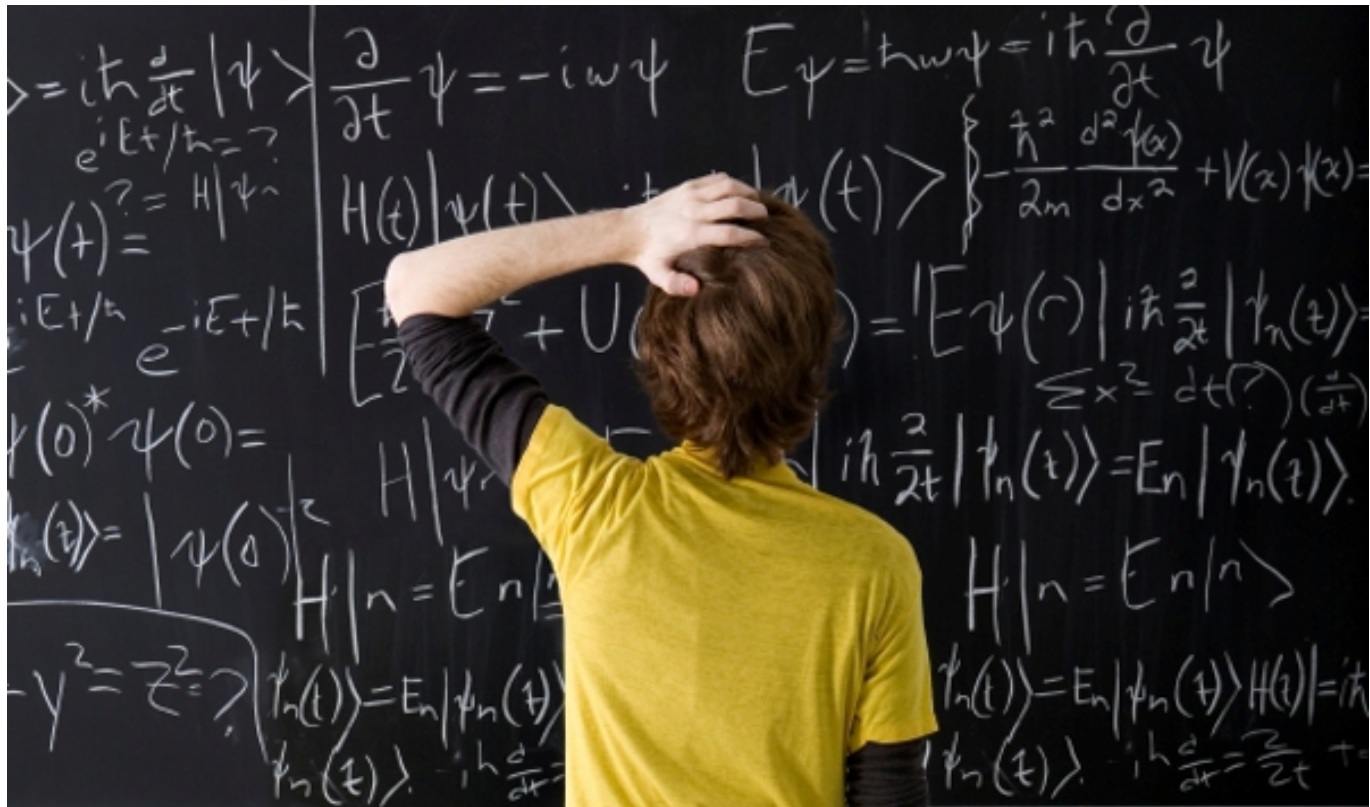
Algoritmi ed Elaboratori

Questo è (parte di) un elaboratore (analytical engine, 1837):



Algoritmi ed Elaboratori

Questo è un elaboratore:



Algoritmi ed Elaboratori

Un **elaboratore** è una entità che può:

- **Memorizzare** informazioni
- Eseguire su di esse alcune **operazioni elementari**

L'informatica è la scienza della **rappresentazione
e dell'**elaborazione** dell'**informazione**.**

L'attività di preparare un algoritmo per un elaboratore:

- Si chiama implementazione o codifica
- Produce come risultato un programma

...Ma rivedremo queste due definizioni tra breve





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fondamenti di Informatica T

Linguaggi di Programmazione

Barriere di Linguaggio

Ci interessano gli **elaboratori elettronici digitali (computer)**

- Informazioni codificate mediante numeri binari (0/1)
- Operazioni elementari piuttosto semplici (specifica binaria)

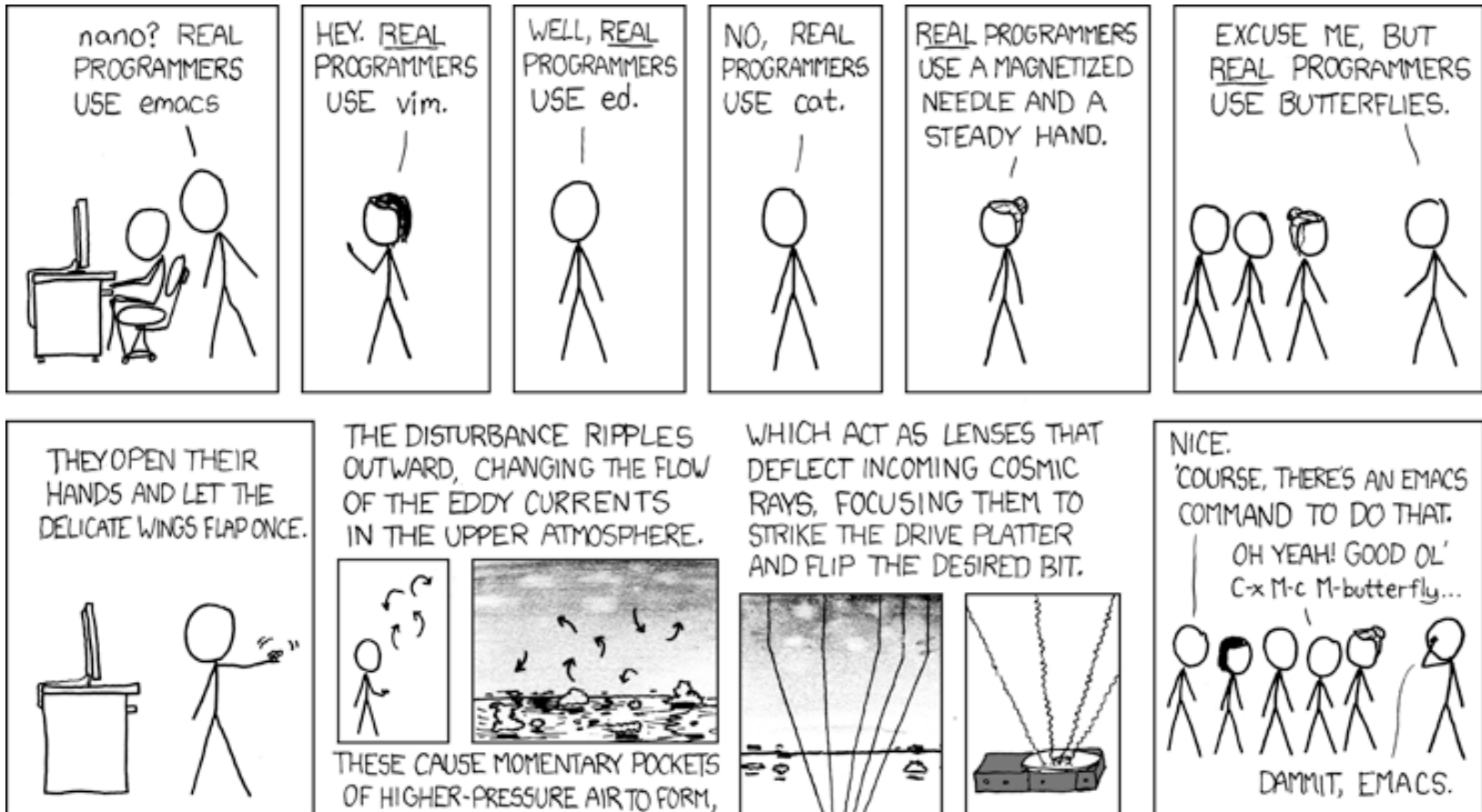
Il linguaggio nativo di un computer si chiama **linguaggio macchina**:

000034f0	54 b5 dd 4d e8 bf 0a 3c c6 5d ee 0b 97 3d b7 11	TpÝMè¿.<E]í.==.
00003500	91 72 e9 f5 96 89 80 3c 9a 0b 31 ce a6 fb f4 f9	`réð-¿E<š.1Î!ûòù
00003510	1c 04 df 70 96 e6 5d c2 b9 65 25 cb 9f 0e e7 23	..ßp-æ]Â'ešEÿ.ç#
00003520	ca 6c 62 8c dd a6 1a db dc 19 d3 f1 fd 06 27 33	ÊlbæÝ!..ÛÛ.Óñý.'3
00003530	a8 21 dc ce a3 53 94 6f 0c 05 8e ad 62 f9 3a b5	~!ÜÎ¿S"o..Ž-bù:p
00003540	d9 df dd e8 a4 b0 49 7d 46 91 85 22 e2 4e 0e 79	ÛßÝè«°I}F'..."âN.y
00003550	ed 6b 61 0d 1a e8 f4 da 6e 1e 20 00 8f 0b 5c 05	íka..èðÚn. . .\.
00003560	db 4f c5 45 09 d6 05 f3 ef c6 e8 78 e1 ef d5 e6	ÛOÂE.Ö.óïEèxáiÏæ
00003570	d7 93 a3 19 16 5d b3 d3 ba 46 38 6a c6 31 81 71	×"¿..]°Ó°F8jÆ1 q
00003580	60 dc bc 56 6a 02 9f 16 34 f7 31 6e 60 02 d2 7a	`Û«Vj.ÿ.4÷1n`.Òz
00003590	fa ba 59 86 b5 48 bf 18 1b 0a 21 66 e6 48 95 15	ú°Y+µH¿...!fæH•.
000035a0	e4 de 1d e7 59 64 59 23 bb cc c5 c0 af 16 4a a4	ãß.çYdY#»ÎÂÂ°.J«
000035b0	f1 a4 79 7e af 45 71 5e 46 04 cf 63 74 c9 6d eb	ñ«y~°Eq^F.ÏctÉmë
000035c0	d8 11 9d 00 d9 c8 31 e7 13 28 8c 01 2a 83 3e 85	Ø. .ÛÈ1ç. (E.*f>...
000035d0	61 9f 98 a4 40 c3 ab c2 06 04 b5 fd 51 52 20 59	aÿ~«@Â«Â..µýQR Y
000035e0	4c 6d 62 f6 6e 1e e6 e8 ba 45 95 cb 89 0c 9a 74	Lmbön.æè°E•E%.št
000035f0	19 cf ee 56 be 30 96 13 87 86 30 8a 24 26 e4 60	.ïiv«0-..+0Š¿æã`



Barriere di Linguaggio

Conseguenza: usare un computer in modo diretto è **difficile**



Linguaggi e Traduttori

Come venirne fuori (mentalmente sani)?

Usare il linguaggio macchina per definire un programma:

- ...Che accetti un **linguaggio più ricco ed astratto**
- ...E lo traduca in **linguaggio macchina**

Un programma del genere si chiama **traduttore:**



Linguaggi e Livelli di Astrazione

Con i traduttori possiamo definire linguaggi sempre più astratti:

Linguaggio **Macchina**:

```
0100 0000 0000 1000
0100 0000 0000 1001
0000 0000 0000 1000
```

In forma binaria; direttamente interpretabile dalla macchina

Linguaggio **Assembler**:

```
LOADA H
LOADB Z
ADD
```

Operazioni corrispondenti 1-1 con le quelle della macchina, ma espresse in forma simbolica

Linguaggi **di alto livello**:

```
int main() {
    int A = 1;
    ...
}
```

Sufficientemente astratti da essere **indipendenti dalla macchina**



Linguaggio di Alto Livello

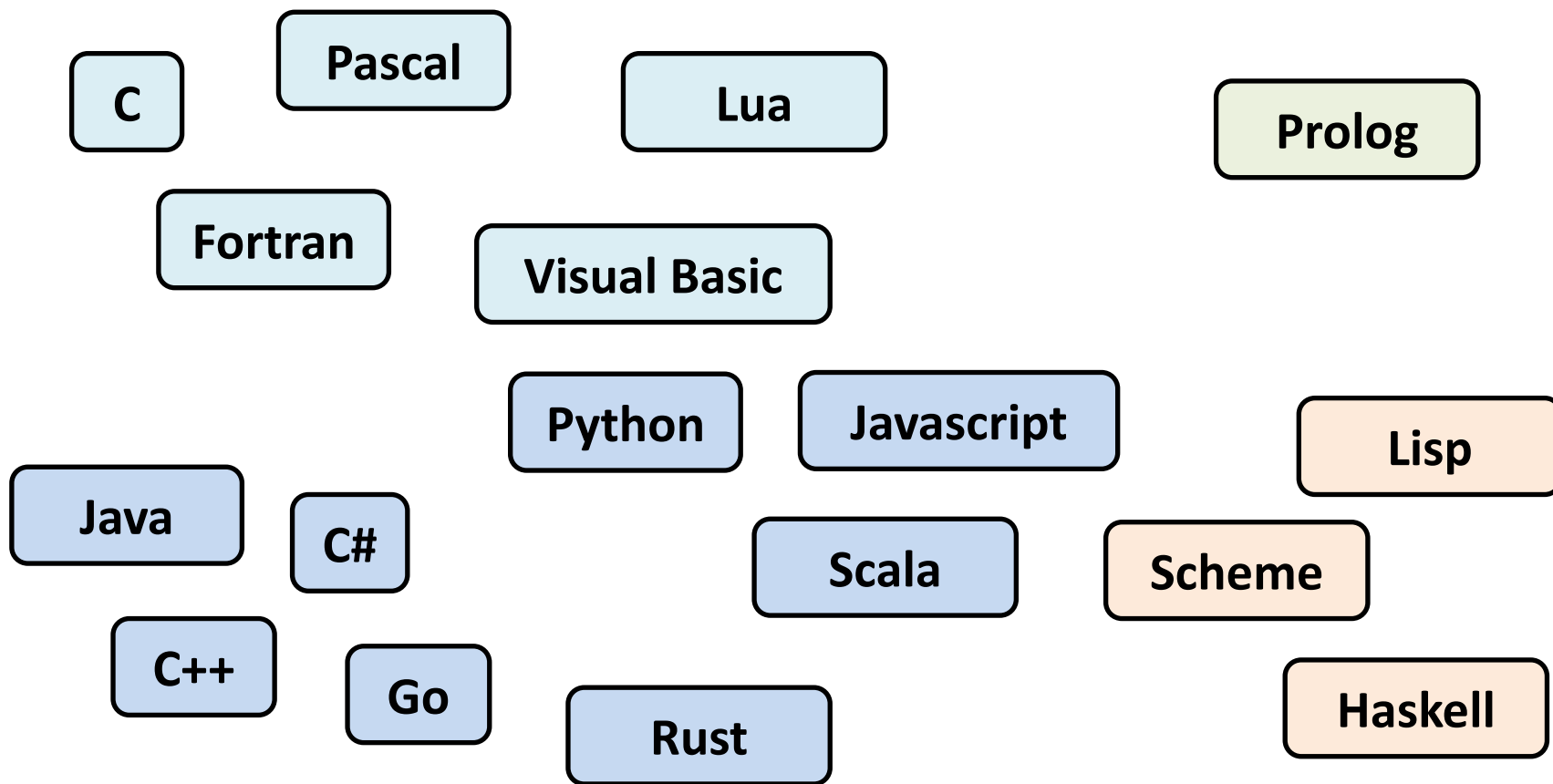
Raggiunto un livello di astrazione sufficiente:

- Un linguaggio si può basare su una **astrazione dell'elaboratore**
- Programmare diventa molto più **facile**
- Lo stesso programma può essere **tradotto su elaboratori diversi**

Un linguaggio (di programmazione) così si dice **di alto livello**



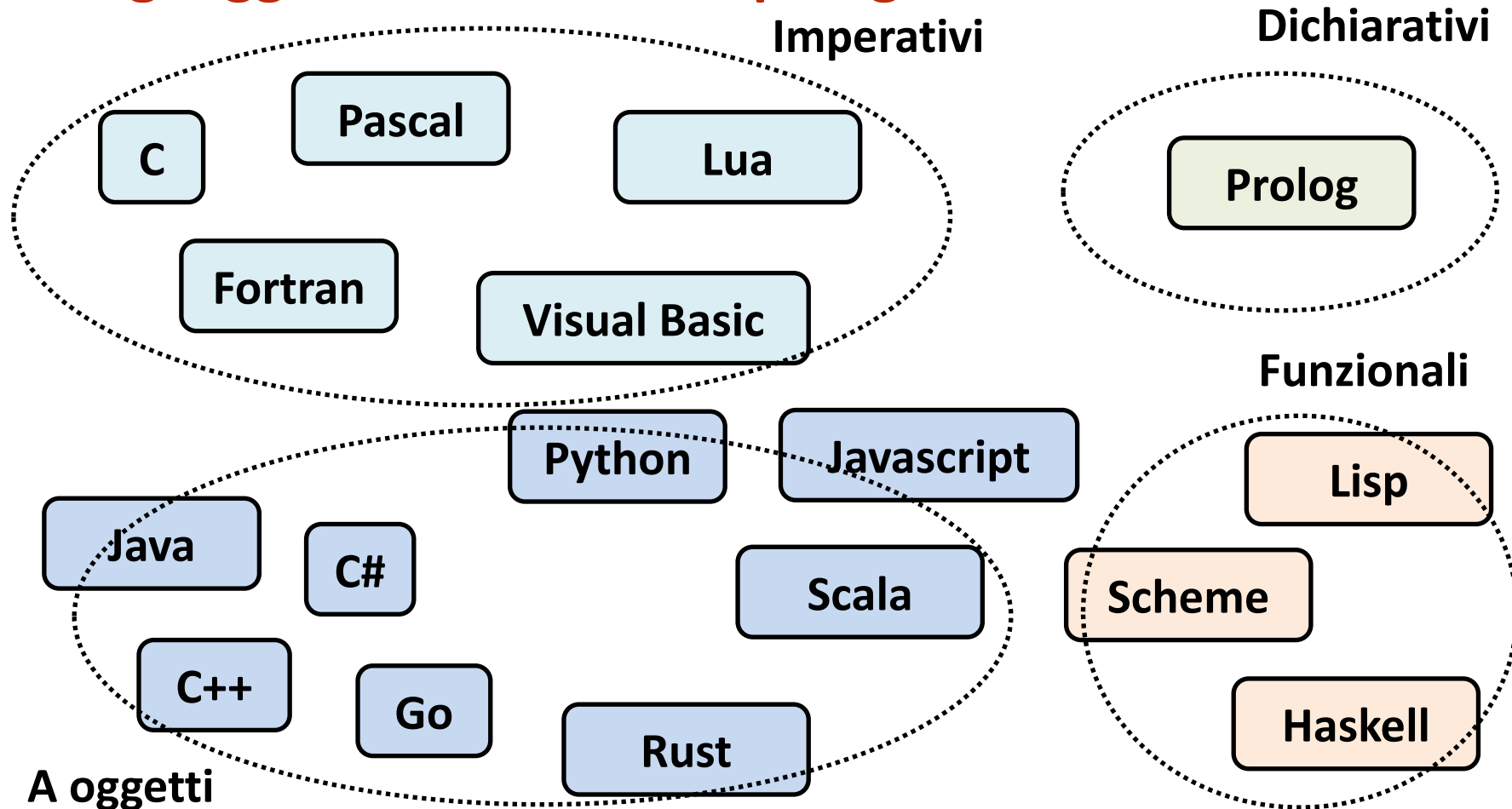
Linguaggi di Alto Livello: Esempi



Elaboratore:



Linguaggi di Alto Livello: Tipologie



Elaboratore:



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi imperativi:

Astrazione: macchina in grado di eseguire sequenze di istruzioni

```
int main() {  
    printf("Hello, World!");  
    return 0;  
}
```



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi a oggetti:

Astrazione: tipi di dati astratti con operazioni programmabili

```
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```



Linguaggi di Alto Livello: Tipologie ed Astrazioni

Linguaggi funzionali:

Astrazione: funzioni matematiche

```
((lambda (x) (+ x x)) 5)
```

Linguaggi logici:

Astrazione: logica matematica

```
likes(mary, food).  
likes(john, wine).  
?- likes(mary, X)
```



Tipi di Traduttori

Distinguiamo due categorie principali di traduttori

...A seconda di quando avvenga l'esecuzione

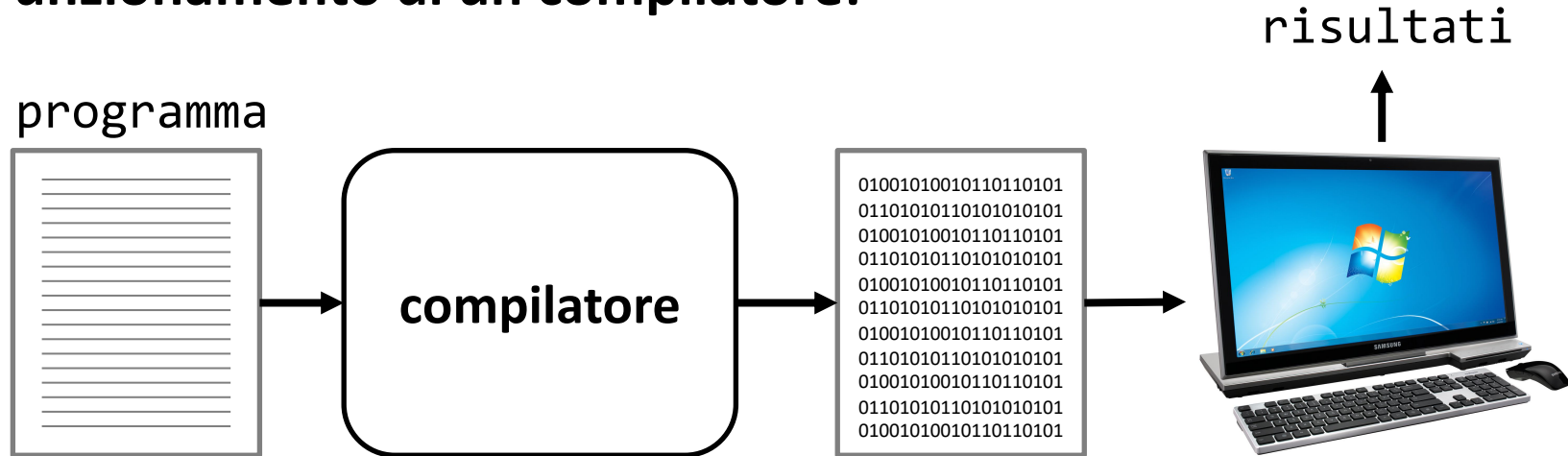
- 1. Compilatori:** l'intero programma viene tradotto (in linguaggio macchina) e quindi eseguito
- 2. Interpreti:** le operazioni vengono tradotte ed eseguite individualmente

In pratica esistono anche gradi intermedi



Tipi di Traduttori

Funzionamento di un compilatore:

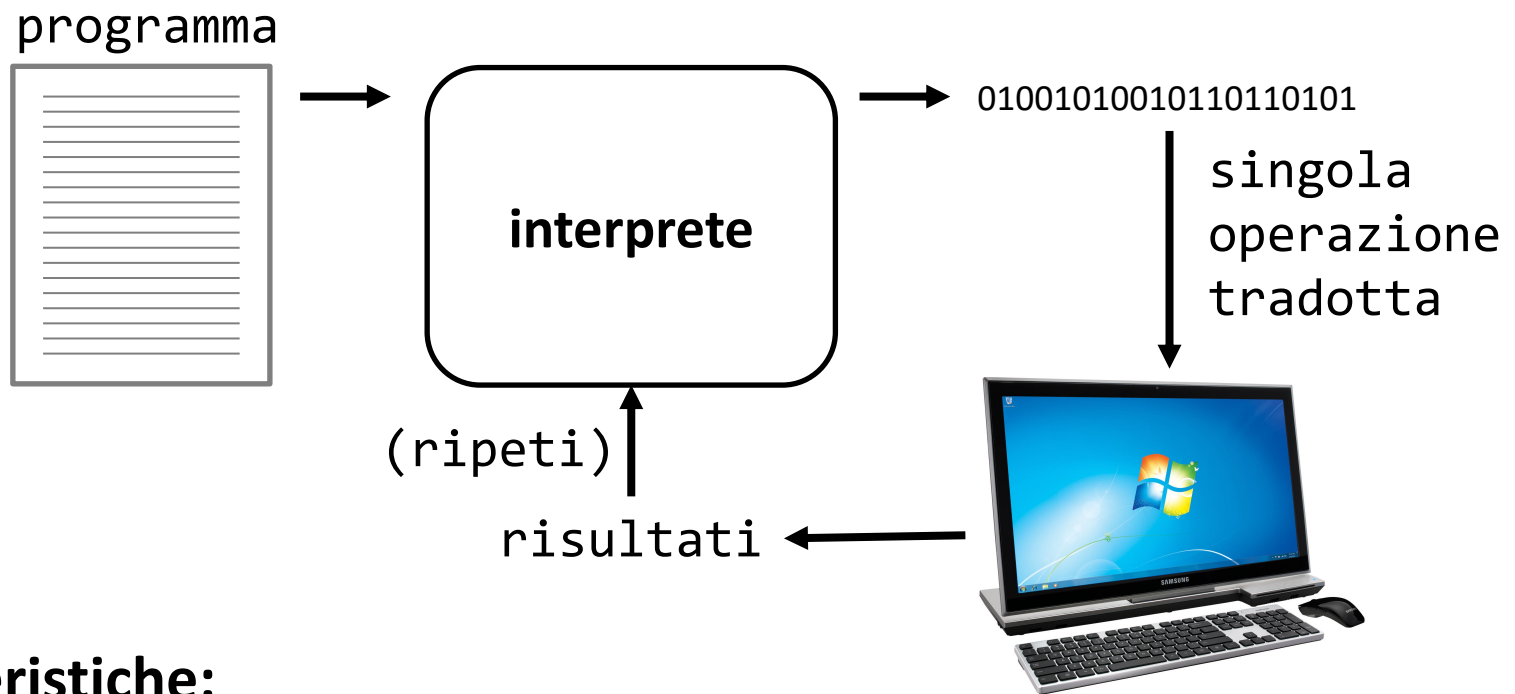


Caratteristiche:

- Esecuzione molto veloce (niente overhead di traduzione)
- Programmazione lenta (l'intera traduzione va ripetuta)

Tipi di Traduttori

Funzionamento di un interprete:



Caratteristiche:

- Esecuzione lenta (overhead di traduzione per ogni operazione)
- Programmazione veloce (no necessità di traduzione)



Che Tipo di Linguaggio?

Cos'è un linguaggio?

“Un linguaggio è un insieme di parole e di metodi di combinazione delle parole usate e comprese da una comunità di persone.”

È una definizione poco precisa:

- Non evita le **ambiguità** dei linguaggi naturali
- Non descrive processi computazionali automatizzabili
- Non aiuta a stabilire proprietà



Che Tipo di Linguaggio?

Ci occorre una definizione di linguaggio più precisa

Un linguaggio come **sistema matematico**

Deve poter rispondere in modo **non ambiguo** a domande come:

- Quali sono le frasi lecite?
- Una determinata frase è valida per il linguaggio?
- Quale è il significato di una frase?
- Quali sono gli elementi linguistici primitivi?

Se definiamo questi aspetti, abbiamo definito sia il **linguaggio** che la **macchina astratta** su cui si basa



Che Tipo di Linguaggio?

In pratica, dobbiamo definire **due aspetti** di un linguaggio:

Sintassi: l'insieme di **regole formali** per la scrittura di programmi in un linguaggio, che dettano le modalità **per costruire frasi corrette**

Semantica: l'insieme dei **significati** da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

Ne conseguono due tipi di errori:

- Errori sintattici: la frase non può essere tradotta
- Errori semantici: la frase non ha il significato corretto



Sintassi

Le regole sintattiche sono espresse attraverso notazioni formali

Qualche esempio importante:

- BNF (Backus-Naur Form)
- EBNF (Extended BNF)
- Diagrammi sintattici



Sintassi EBNF

Sintassi di un numero naturale

```
<naturale> ::=  
    0 | <cifra-non-nulla>{<cifra>}  
<cifra-non-nulla> ::=  
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
<cifra> ::=  
    0 | <cifra-non-nulla>
```



Sintassi EBNF

```
<naturale> ::=  
    0 | <cifra-non-nulla>{<cifra>}
```

Significa che un numero naturale si può riscrivere come 0 oppure (simbolo "|") come una cifra non nulla seguita da zero o più (simbolo "{}") cifre

```
<cifra-non-nulla> ::=  
    1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

una cifra non nulla si può riscrivere come 1 oppure 2 oppure 3...

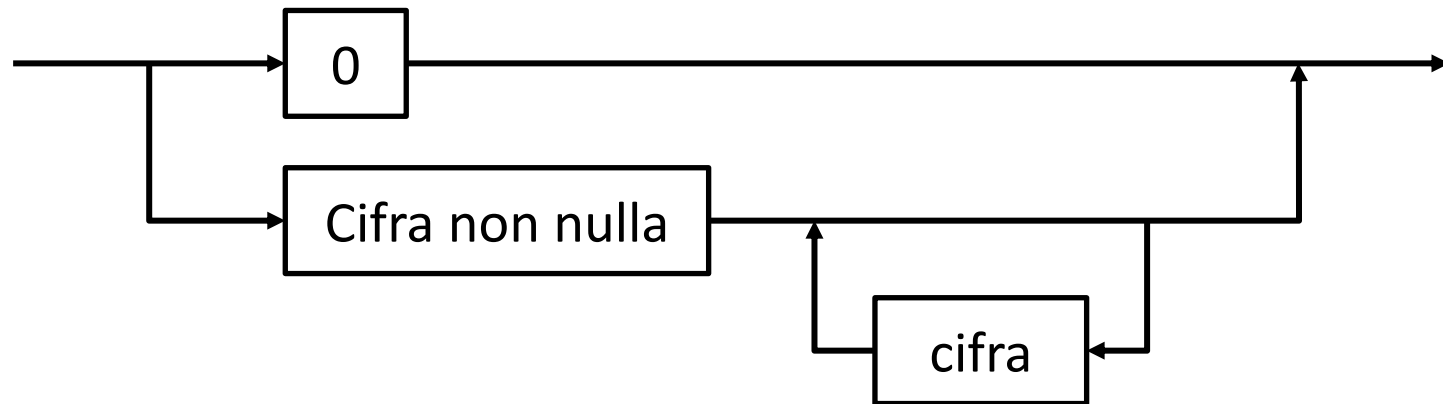
```
<cifra> ::=  
    0 | <cifra-non-nulla>
```

una cifra si può riscrivere come 0 oppure come una cifra non nulla (definita precedentemente)



Diagrammi Sintattici: Esempio

Sintassi di un numero naturale



Semantica

La semantica è esprimibile:

- A parole (poco precisa ed ambigua)
 - ...ma utilizzata in pratica (e.g. documentazione)
- Mediante **azioni**
 - Semantica operazionale
- Mediante **funzioni matematiche**
 - Semantica denotazionale
- Mediante **formule logiche**
 - Semantica assiomatica



Linguaggio e Programma

Possiamo finalmente dare due definizioni più precise:

Linguaggio di Programmazione: una notazione formale con una **semantica eseguibile**.

Programma: un testo scritto in un linguaggio di programmazione

- Un programma è spesso la codifica di un algoritmo
- ...Ma non necessariamente
 - E.g. server web, interfacce utente



Ambiente di Sviluppo

Per programmare quindi ci occorre (oltre ad un elaboratore):

- Un editor (di solito di testo) per il linguaggio
 - E.g. blocco note, VIM, emacs, VS Code...
- Un traduttore
 - Compilatore o interprete (a seconda del linguaggio)
- ...Ed in pratica un gestore di file
 - E.g. esplora risorse (windows), finder (OSX)

Oppure una applicazione che svolga più di uno di questi compiti,
i.e. un **ambiente di sviluppo integrato**





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fondamenti di Informatica T

Introduzione al Linguaggio C

Il Linguaggio C

Un po' di storia

- definito nel 1972 (AT&T Bell Labs) per sostituire l'Assembler
- prima definizione precisa: Kernigham & Ritchie (1978)
- prima definizione ufficiale: ANSI C (1983)
- Successive standardizzazioni:
 - C90, C95, C99, C11, C18



Il Linguaggio C

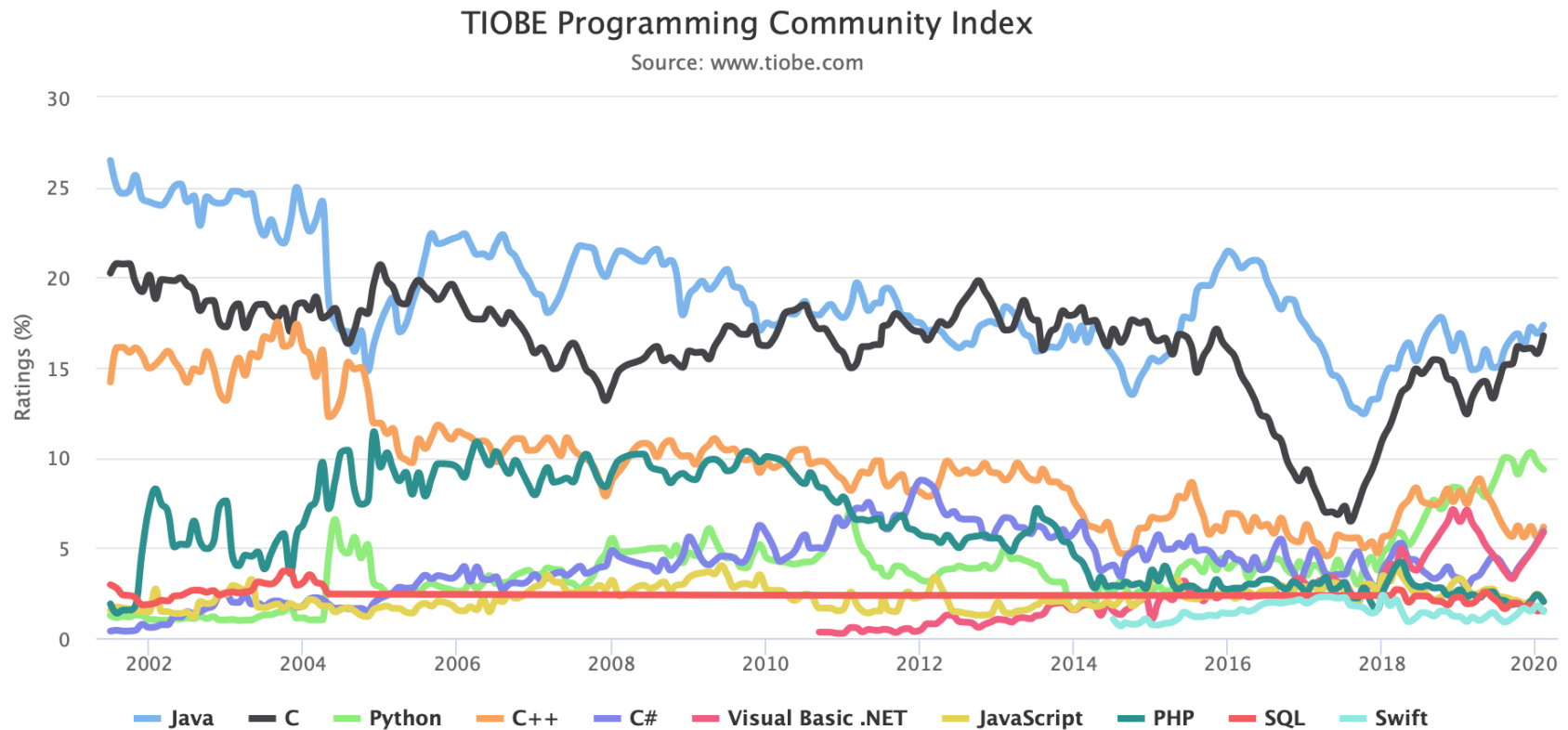
Caratteristiche

- Linguaggio:
 - Imperativo
 - Strutturato a blocchi
- Adatto come **linguaggio di sistema** 😊
 - Per sistemi operativi, compilatori, driver hardware..
- Portabile, **efficiente**, sintetico 😊
 - Supercalcolo, sistemi di controllo, simulazione, videogiochi
- A volte poco leggibile 😞
- Di approccio non immediato 😞



Il Linguaggio C

Datato, ma non in disuso (dati Feb 2020)!



Ambiente di Sviluppo in C

Il linguaggio C adotta un approccio a **compilatore**

La catena di compilazione ha due componenti principali

- Un programma consiste di uno o più file di testo
 - Estensioni “.c” (più avanti vedremo anche “.h”)
- Questi vengono elaborati individualmente da un **compilatore**
 - I file risultati si dicono “file oggetto” (estensione “.o”)
 - Contengono codice macchina, più dei “segnaposto” nei punti in cui sono utilizzate funzionalità descritte in altri file
- Un secondo componente, detto **linker**, risolve tali dipendenze e produce un file eseguibile (su windows estensione “.exe”)



Un Esempio di Codice C

File “main.c” (non preoccupatevi di capire, per adesso):

```
#include <stdio.h>

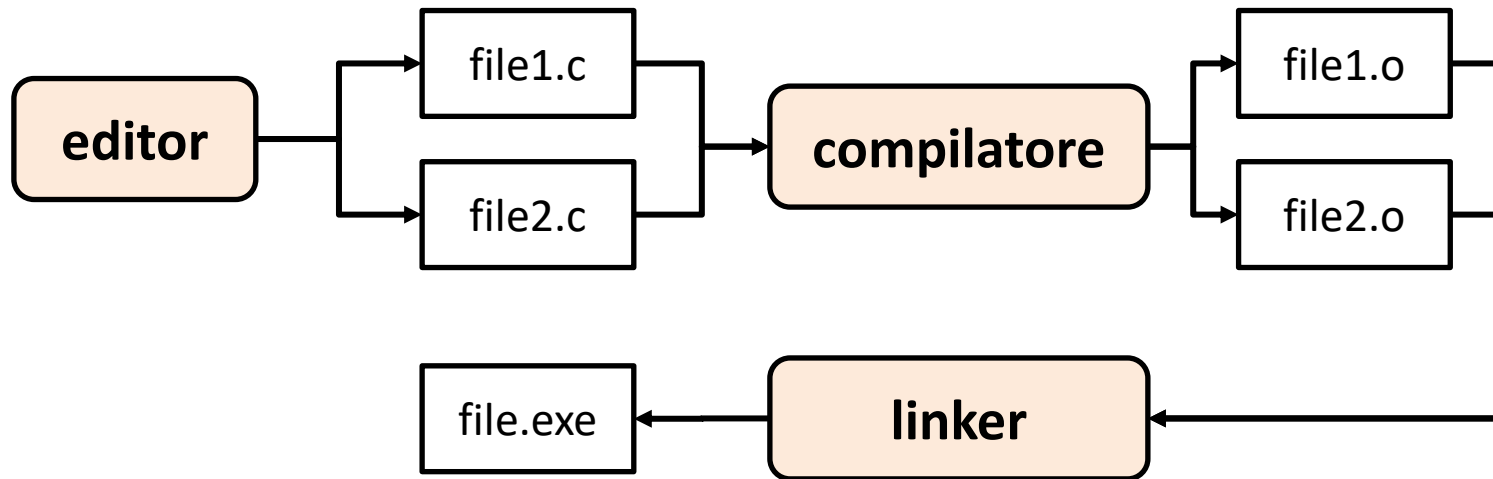
int fact(int n) {
    if (n == 0) return 1;
    else return n * fact(n-1);
}

int main() {
    int n = 7;
    int y = fact(n);
    printf("Fattoriale di %d: %d\n", n, y);
}
```



Ambiente di Sviluppo in C

Fasi della trasformazione di un programma C in un file eseguibile



- Quando il programma viene modificato, le fasi vanno ripetute
- Vedremo più avanti un terzo **componente** (preprocessore)



Ambiente di Sviluppo in C

- Il compilatore individua automaticamente gli errori sintattici
- Per gli errori **semantici** si può utilizzare un **debugger**

Il debugger è un altro componente dell'ambiente di sviluppo C

Per usare un debugger, in fase di compilazione:

- Si inseriscono nel file eseguibile dei “marcatori”
- Ogni marcatore fa riferimento ad una riga del programma

In fase di esecuzione, il vero e proprio debugger (un programma):

- Può interrompere l'eseguibile in corrispondenza dei marcatori
- Permette di ispezionare una serie di informazioni utili



Ambiente di Sviluppo in C

Per programmare occorre quindi un compilatore (& un linker)

Qualche esempio importante per il corso:

- Compilatore **gcc**
 - Disponibile su linux
 - Disponibile su windows (mediante il pacchetto MinGW)
- Compilatore **clang**
 - Disponibile su linux e OSX
 - Replica le funzionalità di gcc (i due sono compatibili)

Se lo si desidera, si può usare un debugger

- Esempi: gdb (linux e windows), LLDB (OSX)



Ambiente di Sviluppo in C

Per programmare **occorre quindi un editor**

Di per sé un editor di testo è sufficiente:

- Blocco note
- Editor specializzati per codice: Notepad++, VIM, emacs, VS Code, Atom, Sublime Text...

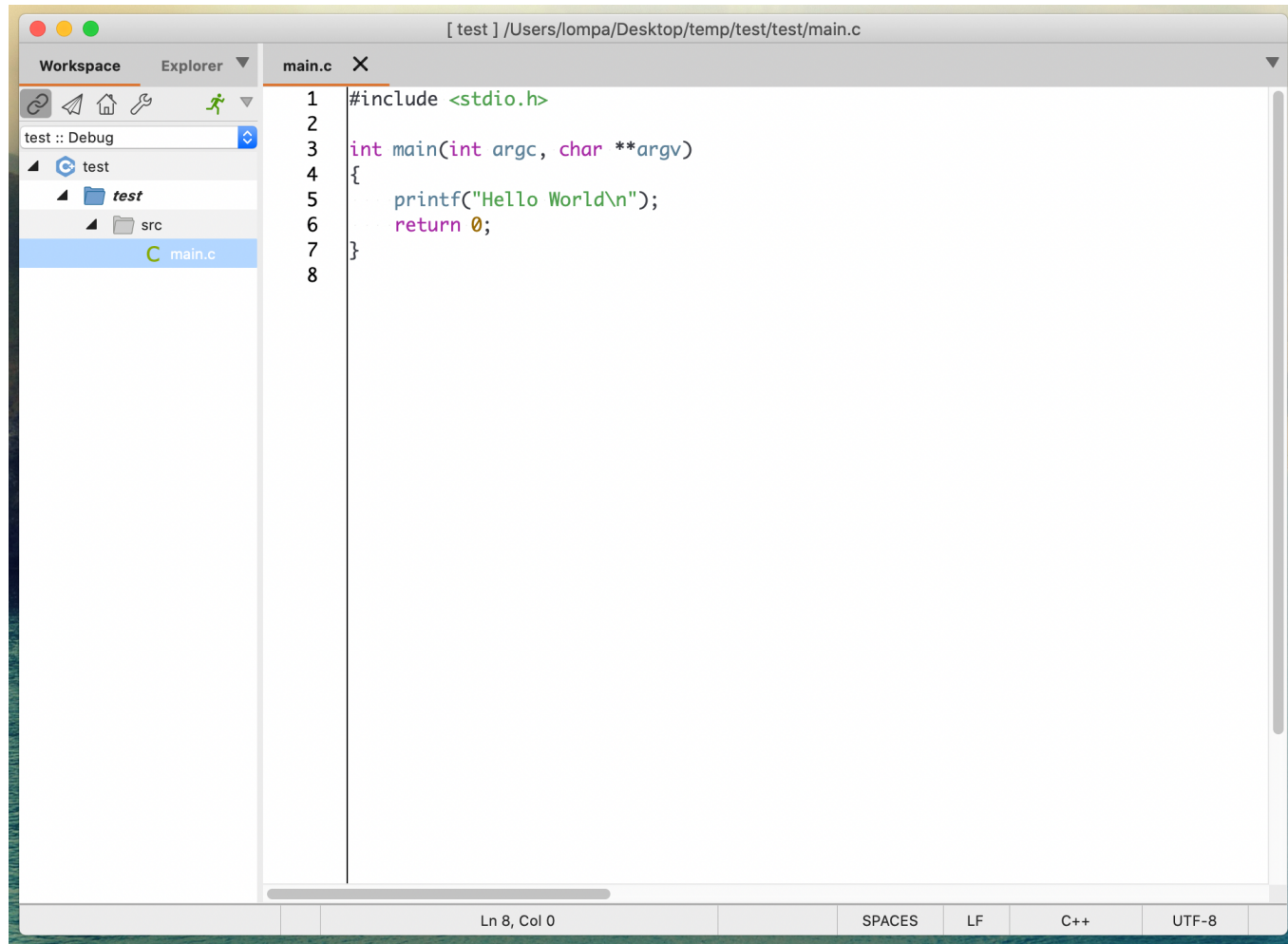
Può però fare comodo un **ambiente di sviluppo integrato (IDE)**

- Svolge le funzioni dell'editor, facilita l'utilizzo del compilatore e del debugger, semplifica la navigazione tra i vari file...
- Qualche esempio: Visual Studio, Xcode, DevCpp...
- Per questo corso: **CodeLite**



Ambiente di Sviluppo in C

Interfaccia di CodeLite (lo vedrete meglio in laboratorio)





ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fondamenti di Informatica T

Andrea Acquaviva <andrea.acquaviva@unibo.it>

Michele Lombardi <michele.lombardi2@unibo.it>

Andrea Borghesi <andrea.borghesi3@unibo.it>

Giuseppe Tagliavini <giuseppe.tagliavini@unibo.it>

Allegra De Filippo <allegra.defilippo@unibo.it>