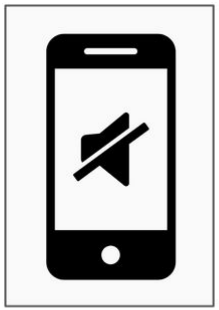
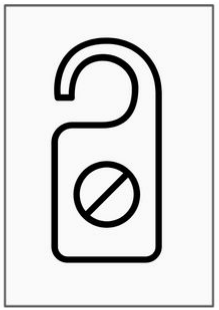
La lezione sta per iniziare

Mentre aspettiamo elimina le distrazioni



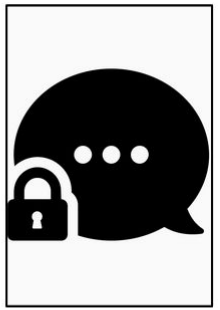
Telefono muto

Porta chiusa



Cartello appeso

E-mail chiusa



Social chiusi

Regole di interazione

1. Mettere il microfono in modalità “muto” e disattivare la webcam
2. Prenotarsi sulla chat per fare eventuali domande
3. Abilitare temporaneamente il microfono quando richiesto dal docente

Wrap-up

Informatica di base – a.a. 2019/2020

Silvio Peroni
0000-0003-0530-4305

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia
silvio.peroni@unibo.it – @essepuntato – https://www.unibo.it/sitoweb/silvio.peroni/

DIPARTIMENTO DI FILOLOGIA CLASSICA E ITALIANISTICA

Approssimazione della soluzione di Turing

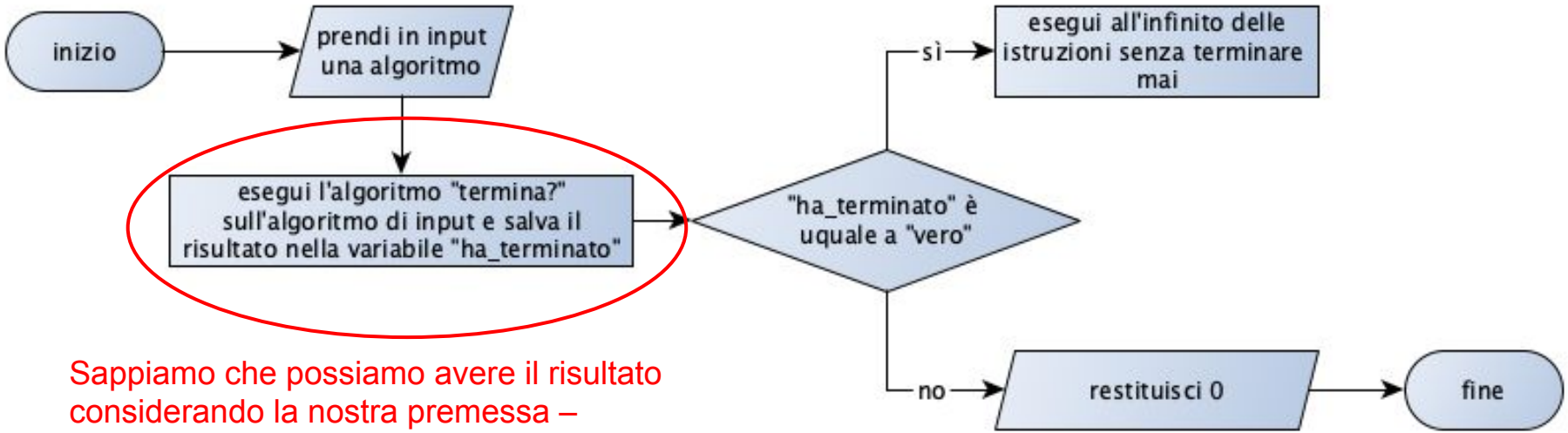
Premessa: supponiamo **sia possibile sviluppare** l'algoritmo "termina?", che prende in **input un certo algoritmo** e restituisce **"vero"** nel caso in cui l'algoritmo specificato come **input termina**, mentre restituisce **"falso" in caso contrario**

NB: è soltanto un algoritmo ipotetico, stiamo supponendo che possiamo svilupparlo in qualche modo, senza mostrare come farlo davvero

Usiamo questo algoritmo per crearne un altro

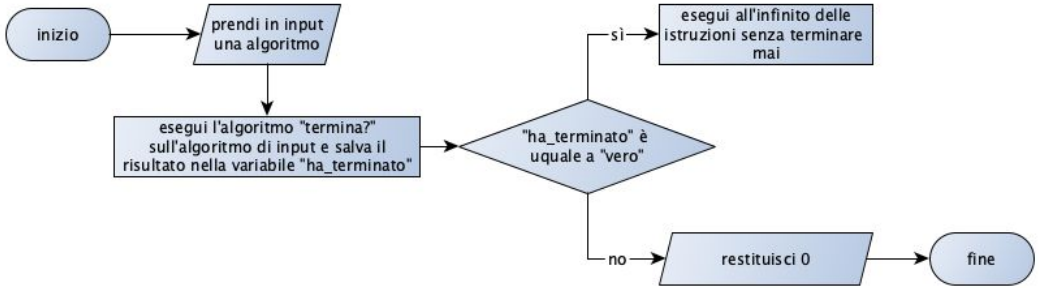
Nuovo algoritmo

Nuovo algoritmo: prendi in input un algoritmo e restituisci 0 se l'algoritmo in input termina, mentre non terminare in caso contrario



Sappiamo che possiamo avere il risultato considerando la nostra premessa – ovvero che "ha_terminato" esiste

E se usiamo il nuovo algoritmo come suo input?



Caso 1: se l'algoritmo "termina?" afferma che il nuovo algoritmo **termina**, conseguentemente (per come è definito) il nuovo algoritmo **non termina** l'esecuzione

Caso 2: se l'algoritmo "termina?" afferma che il nuovo algoritmo **non termina**, e conseguentemente (per come è definito) il nuovo algoritmo restituisce 0 e **termina** l'esecuzione

Paradosso: l'algoritmo che verifica se un altro termina **non può esistere**

Grammatica formale

Un insieme di regole di produzione di forma `premessa ::= espressione`, dove la `premessa` e l'`espressione` possono contenere uno o più:

- simboli **terminali** (specificati tra virgolette), che identificano tutti i simboli elementari del linguaggio in considerazione (come nomi, verbi, etc.)
- simboli **non terminali** (specificati tra parentesi angolari), che identificano tutti i simboli di una grammatica formale che possono essere sostituiti da una combinazione di simboli terminali e non terminali

Esempio:

```
<frase> ::= "Io" <verbo>
<verbo> ::= "scrivo" | "leggo"
```

Il simbolo non terminale `<frase>` è il simbolo iniziale.
Frasi che possiamo produrre da questa grammatica:

- Io scrivo
- Io leggo

Gerarchia di Chomsky

Chomsky ha proposto una gerarchia per descrivere formalmente le relazioni che possono esistere tra diverse grammatiche in termini delle loro possibili strutture sintattiche che sono in grado di generare

Queste tipologie sono caratterizzate dal tipo di simboli che possono essere usati nella `premessa` e nell'`espressione` delle regole di produzione (lettere greche usate per indicare una combinazione di simboli terminali e/o non terminali):

- grammatiche **regolari** – `<nt> ::= "t" | "t" <nt>`
- grammatiche **libere dal contesto** – `<nt> ::= γ`
- grammatiche **dipendenti dal contesto** – `α <nt> β ::= α γ β`
- grammatiche **ricorsivamente enumerabili** – `α ::= β`

Grammatiche dipendenti dal contesto

α <nt> β ::= α γ β

Esempio:

```
<sentence> ::= <subject pronoun> <verbphrase>
<subject pronoun> ::= "I"
<verbphrase> ::= <verb> <noun>
"I" <verb> <noun> ::= "I" "read" "a" <noun>
<noun> ::= "book"
```

Grammatiche ricorsivamente enumerabili

α ::= β

Esempio:

```
<sentence> ::= <subject pronoun> <verbphrase>
<subject pronoun> ::= "I"
<verbphrase> ::= <verb> <noun>
"I" <verb> <noun> ::= "I" "read" "a" "book"
```

Non demoniaci, ma mitologici

Nel destro lato è Scilla; nel sinistro È l'ingorda
Cariddi. Una vorago
D'un gran baratro è questa, che tre volte
I vasti flutti rigirando assorbe,
E tre volte a vicenda li ributta
Con immenso bollor fino a le stelle.

Virgilio, Eneide

Dipinto: Odysseus in front of Scylla and Charybdis
Autore: Johann Heinrich Füssli

https://it.wikipedia.org/wiki/File:Johann_Heinrich_F%C3%BCssli_054.jpg

Linguaggio macchina

Un insieme di istruzioni che possono essere eseguite direttamente dalla CPU (*central processing unit*, o processore) di un computer elettronico

Basato sul **codice binario** – una sequenza di 0 e 1 – rivisitato da Leibniz alla fine del diciassettesimo secolo

Gottfried Leibniz

```
11010011101110110011011011111110010110110111000011110100110100111000111100001

11010011101110110011011011111110010110110111000011110100110100111000111100001

   i     n     f     o     r     m     a     t     i     c     a
```

Linguaggi di programmazione a basso livello

Forniscono **un livello di astrazione** sopra il linguaggio macchina

Permettono di scrivere programmi in modo che siano un pochino più intellegibili dagli umani

Il più famoso linguaggio di questo tipo è l'Assembly – anche se introduce simboli più comprensibili, di solito una linea di codice in Assembly rappresenta una specifica istruzione in linguaggio macchina

```
fib:
        mov edx, [esp+8]
        cmp edx, 0
        ja @f
        mov eax, 0
        ret

        @@:
        cmp edx, 2
        ja @f
        mov eax, 1
        ret

        @@:
        push ebx
        mov ebx, 1
        mov ecx, 1

…
```

Linguaggi di programmazione ad alto livello

Caratterizzati da un **forte livello di astrazione** dal linguaggio macchina – esempio: Python

Possono usare parole proprie del linguaggio naturale per definire costrutti specifici, così da essere di più facile comprensione per un umano

Più astrazione da un linguaggio di programmazione a basso livello è fornita, più comprensibile è il linguaggio

```
def fib(n):
    if n <= 0:
        return 0
    elif n <= 2:
        return 1
    else:
        a = 1
        b = 1
        while True:
            c = a + b
            if n <= 3:
                return c
            a = b
            b = c
            n = n - 1
```

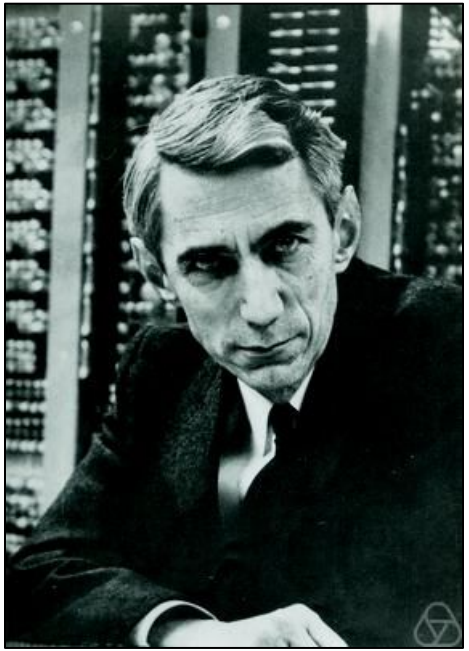
Byte e bit

Byte: unità minima di informazione occupabile su un computer, che storicamente corrisponde al numero di bit necessari per codificare un carattere sul computer

Bit (contrattura di *binary digit*): l'unità minima di informazione che si può scambiare in una comunicazione e può assumere solo uno di due valori: 0 o 1

1 byte = 8 bit

Il concetto di bit è stato usato da diversi studiosi del passato (ad esempio Babbage con le schede perforate), ma introdotto come termine formalmente da Claude Shannon (1948)



Teoria dell’informazione

Nell’articolo che introduce il bit, Shannon in realtà mette le basi di uno specifico campo di ricerca e studio chiamato **teoria dell’informazione**

La teoria dell’informazione si occupa di studiare come quantificare, memorizzare, e scambiare informazione, che ha tuttora svariate applicazioni pratiche, oltre che aver caratterizzato e veicolato l’invenzione di diverse tecnologie del passato, incluso lo sviluppo di Internet

Esempio: evincere la capacità massima di un canale per trasmettere informazione in modo affidabile è stato uno degli studi più importanti della teoria dell’informazione (a cui l’MTU è strettamente connesso)

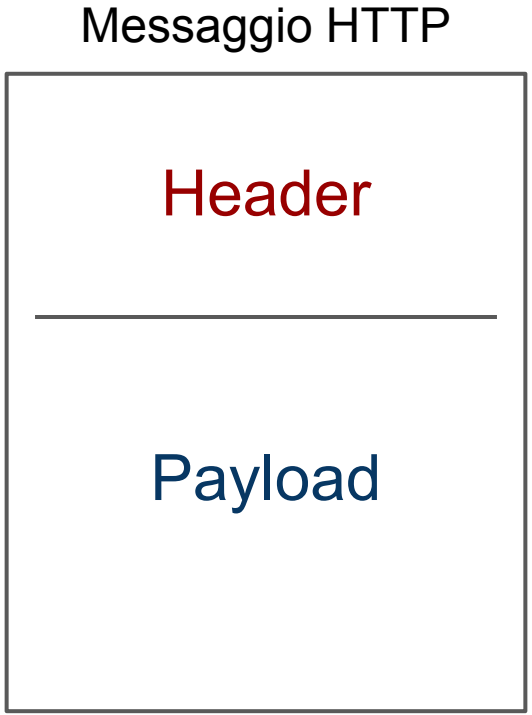
Da bit a simboli e viceversa

Ogni carattere, numero, programma, applicazione all'interno di un computer, così come un qualunque messaggio da scambiare in Internet, è **codificato** come una sequenza di bit che, in qualche modo, vengono poi **decodificati** con la sequenza di caratteri corretta dalla particolare applicazione che riceve ed interpreta quell'informazione

Per esempio, nella codifica binaria tradizionale dei numeri interi, il numero 0 è rappresentato dalla sequenza "0", il numero 1 da "1", il numero 2 da "10", il numero 3 da "11", il numero 4 da "100", e così via

Messaggio HTTP

i metadati relativi alla comunicazione – come il nome di client usato per effettuare la richiesta, il nome del server web che dovrebbe avere a disposizione la risorsa, etc.



eventuali dati (o contenuto) da accompagnare al messaggio – può essere vuoto nel caso in cui si richiede una pagina, per esempio

HTML

L'**Hypertext Markup Language** (HTML) è il linguaggio di markup usato per creare tutte le **pagine web e le applicazioni** presenti sul Web

È un linguaggio che segue una **sintassi simile a quella XML**, e che mette a disposizione uno specifico vocabolario di elementi ed attributi per identificare i vari ruoli strutturali e semantici di una pagina web

Ogni qual volta viene fatta una richiesta per una pagina web, viene restituita una copia di un documento HTML contenente opportuni marcatori che il **browser è in grado di interpretare e visualizzare** a video in qualche modo

Prima versione di HTML (di Berners-Lee) è del 1990, ed era basata su SGML

Presentazione di un documento HTML: CSS

Nel 1996 viene rilasciata per la prima volta la specifica del Cascading Style Sheet (CSS), ovvero un ulteriore linguaggio che permette di definire il livello presentazionale degli elementi di HTML, come ad esempio:

- il colore e la dimensione del testo contenuto nei titoli e nei paragrafi
- la posizione di sezioni, articoli, immagini o video
- l'animazione dinamica quando si posiziona il cursore del mouse su una voce del menu e/o sui link

```
p {
    text-align: justify;
    color: grey;
}
```

Wrap-up

Le tecnologie informatiche nelle scienze umane

Informatica di base – a.a. 2019/2020

Silvio Peroni
0000-0003-0530-4305

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia
silvio.peroni@unibo.it – @essepuntato – https://www.unibo.it/sitoweb/silvio.peroni/

DIPARTIMENTO DI FILOLOGIA CLASSICA E ITALIANISTICA