Corso di Linguaggi di Programmazione + Laboratorio

Capitolo 1 - Introduzione

Si ringrazia il Dott. Marco de Gemmis per la collaborazione nella predisposizione del materiale didattico

Apprendimento di un linguaggio di programmazione

- Perché un insegnamento “Linguaggi di Programmazione”? Non sarebbe stato sufficiente quello di “Programmazione”?

Apprendimento di un linguaggio di programmazione

- Una delle competenze fondamentali di un *buon* informatico è quella di apprendere un nuovo linguaggio di programmazione con naturalezza e velocità
- Questa competenza non la si acquisisce soltanto imparando *ex novo* molti linguaggi diversi
  - conoscere tanti linguaggi di programmazione oppure conoscere i fondamenti che sono alla base dei linguaggi di programmazione?

Apprendimento di un linguaggio di programmazione

- Lingue “naturali”
  - esistono analogie, somiglianze derivanti dalla genealogia
  - Es.: italiano – rumeno
    - cioccolata = ciocolata
    - insalata = salata
    - treno = tren
- Linguaggi di programmazione
  - è difficile conoscerne un gran numero in modo approfondito
  - è possibile conoscerne a fondo i meccanismi che ne ispirano il *progetto* e l’*implementazione*

Lo studio degli aspetti generali dei linguaggi di programmazione è un passaggio chiave della formazione universitaria e professionale di un informatico

Macchine Astratte e Implementazione dei linguaggi di programmazione

- Qual è il significato dell'espressione: "implementazione di un linguaggio di programmazione"?
- Si tratta di un concetto strettamente correlato a quello di:

MACCHINA ASTRATTA

Macchine Astratte e Implementazione dei linguaggi di programmazione

- Calcolatore = macchina fisica
  - consente di eseguire algoritmi opportunamente formalizzati perché siano "comprensibili" all'esecutore
  - la formalizzazione consiste nella codifica degli algoritmi in un certo linguaggio $\mathcal{L}$ definito da una specifica sintassi
  - la sintassi di $\mathcal{L}$ permette di utilizzare determinati costrutti per comporre programmi in $\mathcal{L}$
  - Un programma in $\mathcal{L}$ è una sequenza di istruzioni del linguaggio $\mathcal{L}$
- Macchina astratta = astrazione del concetto di calcolatore fisico

Definizioni di macchina astratta e linguaggio macchina

- Macchina astratta per $\mathcal{L}$
  - Un insieme di algoritmi e strutture dati che permettono di memorizzare ed eseguire programmi scritti in $\mathcal{L}$
  - Si denota con $\mathcal{M}_{\mathcal{L}}$
  - E' composta da:
    - una memoria per immagazzinare dati e programmi
    - un interprete che esegue le istruzioni contenute nei programmi
- Linguaggio Macchina
  - Data una macchina astratta $\mathcal{M}_{\mathcal{L}}$,il linguaggio $\mathcal{L}$ "compreso" dall'interprete di $\mathcal{M}_{\mathcal{L}}$ è detto linguaggio macchina di $\mathcal{M}_{\mathcal{L}}$

Le operazioni dell'Interprete

- Operazioni per l'elaborazione dei dati primitivi
  - numeri interi, reali
  - operazioni aritmetiche
- Operazioni e strutture dati per il controllo della sequenza di esecuzione
  - servono per gestire il flusso di controllo delle istruzioni
  - strutture dati per memorizzare l'indirizzo della prossima istruzione
  - operazioni per manipolare le strutture dati
    - es.: calcolo dell'indirizzo della prossima istruzione

Le operazioni dell'Interprete

- Operazioni e strutture dati per il controllo del trasferimento dei dati
  - gestiscono il trasferimento dei dati dalla memoria all'interprete
    - es.: recupero degli operandi
  - possono far uso di strutture dati ausiliarie
    - es.: pila
- Operazioni e strutture dati per la gestione della memoria
  - relative all'allocazione di memoria per dati e programmi

Ciclo di esecuzione del generico interprete

1. Acquisisci prossima istruzione da eseguire
2. Decodifica
   - determina l'operazione richiesta dall'istruzione e gli operandi
3. Preleva gli operandi ed esegui l'operazione richiesta
4. Memorizza l'eventuale risultato
5. Torna al punto 1 a meno che l'operazione non sia un HALT

Realizzazione di una macchina astratta

- Per essere effettivamente utilizzata, $\mathcal{M}_{\mathcal{L}}$ dovrà prima o poi utilizzare qualche dispositivo fisico
  - realizzazione “fisica” in hardware
    - algoritmi di $\mathcal{M}_{\mathcal{L}}$ realizzati direttamente mediante dispositivi fisici
- Possiamo pensare a realizzazioni che usino livelli intermedi tra $\mathcal{M}_{\mathcal{L}}$ ed il dispositivo fisico
  - **simulazione mediante software**
  - emulazione mediante firmware
    - microprogrammi in linguaggi di basso livello

Realizzazione mediante software

- Algoritmi e strutture dati di $\mathcal{M}_{\mathcal{L}}$ realizzati in un altro linguaggio $\mathcal{L}'$ già implementato
- $\mathcal{M}_{\mathcal{L}}$ è realizzata mediante programmi in $\mathcal{L}'$ che simulano le funzionalità di $\mathcal{M}_{\mathcal{L}}$
- $\mathcal{M}_{\mathcal{L}}$ è realizzata attraverso la macchina $\mathcal{M}'_{\mathcal{L}'}$
- $\mathcal{M}'_{\mathcal{L}'}$ è detta *macchina ospite* e si denota con $\mathcal{M}o_{\mathcal{L}o}$
- Intuitivamente l’implementazione di $\mathcal{L}$ sulla macchina ospite avviene mediante una qualche “*traduzione*” di $\mathcal{L}$ in $\mathcal{L}o$
- A seconda di come avvenga la traduzione di parla di:
  - **implementazione interpretativa**
  - **interpretazione compilativa**

Definizione di interprete

In generale un programma scritto in $\mathcal{L}$ si può vedere come una funzione parziale: $\mathcal{P}^{\mathcal{L}} : \mathcal{D} \to \mathcal{D}$ t.c. $\mathcal{P}^{\mathcal{L}}(Input) = Output$

Possiamo dare la seguente definizione di interprete di $\mathcal{L}$ in $\mathcal{L}o$:

$$I_{\mathcal{L}}^{\mathcal{L}o} : (\mathcal{P}rog^{\mathcal{L}} \times D) \to D \qquad \text{tale che}$$

$$I_{\mathcal{L}}^{\mathcal{L}o}(\mathcal{P}^{\mathcal{L}}, Input) = \mathcal{P}^{\mathcal{L}}(Input)$$

Non vi è traduzione esplicita dei programmi scritti in $\mathcal{L}$, ma solo un procedimento di decodifica

L'interprete, per eseguire un'istruzione *i* di $\mathcal{L}$, le fa corrispondere un insieme di istruzioni di $\mathcal{L}o$. Tale decodifica non è una traduzione esplicita poiché il codice corrispondente a *i* è eseguito direttamente e non prodotto in uscita

Definizione di compilatore

Un compilatore da $\mathcal{L}$ a $\mathcal{L}o$ è un programma che realizza la funzione:

$$C_{\mathcal{L},\mathcal{L}o} : \mathcal{P}rog^{\mathcal{L}} \to \mathcal{P}rog^{\mathcal{L}o}$$

$$C_{\mathcal{L},\mathcal{L}o}(\mathcal{P}^{\mathcal{L}}) = \mathcal{P}c^{\mathcal{L}o} \qquad \mathcal{P}^{\mathcal{L}}(\mathcal{D}) = \mathcal{P}c^{\mathcal{L}o}(\mathcal{D})$$

Traduzione esplicita dei programmi scritti in $\mathcal{L}$ in programmi scritti in $\mathcal{L}o$

$\mathcal{L}$ linguaggio sorgente  $\mathcal{L}o$ linguaggio oggetto

Per eseguire $\mathcal{P}^{\mathcal{L}}$ sull'input $\mathcal{D}$, bisogna eseguire $C_{\mathcal{L},\mathcal{L}o}$ con $\mathcal{P}^{\mathcal{L}}$ come input. Si avrà come risultato un programma compilato $\mathcal{P}c^{\mathcal{L}o}$ scritto in $\mathcal{L}o$, che sarà eseguito su $\mathcal{M}o_{\mathcal{L}o}$ con il dato di input $\mathcal{D}$

Interpretazione vs. Compilazione

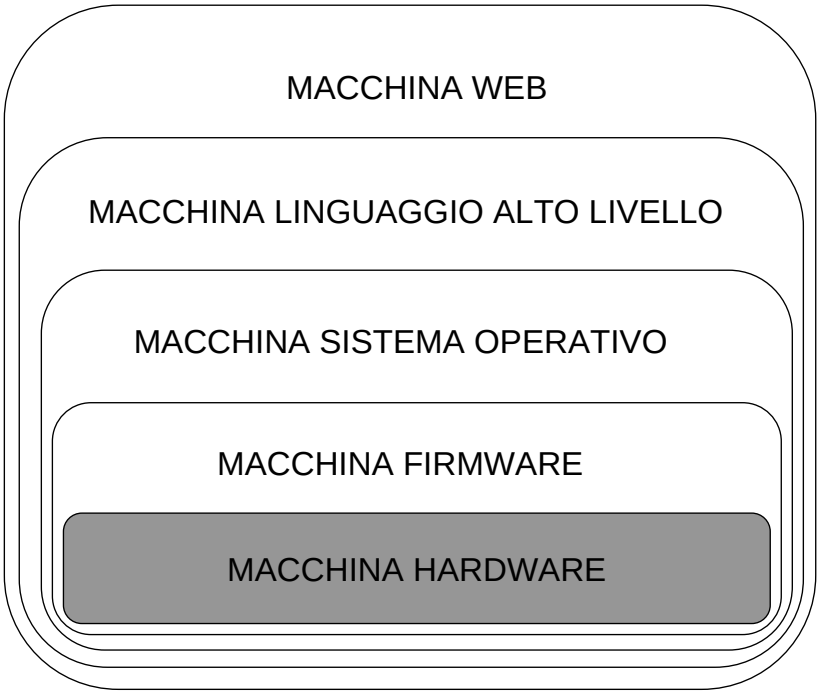
- Interpretazione
  - scarsa efficienza
  - maggiore flessibilità
- Compilazione
  - maggiore efficienza: la decodifica di un'istruzione è fatta una sola volta indipendentemente da quante volte è eseguita
  - minore flessibilità: perdita di informazioni rispetto al programma sorgente (difficile tracciare errori e dunque maggiori difficoltà nel debugging)

In realtà i due approcci sono spesso combinati

Gerarchie di Macchine Astratte

- Possiamo pensare ad una gerarchia di macchine astratte $\mathcal{M}_{\mathcal{L}0}$, $\mathcal{M}_{\mathcal{L}1}$, … $\mathcal{M}_{\mathcal{L}n}$
- $\mathcal{M}_{\mathcal{L}i}$ è implementata sfruttando il linguaggio della macchina sottostante $\mathcal{M}_{\mathcal{L}i-1}$
- $\mathcal{M}_{\mathcal{L}i}$ fornisce a sua volta il proprio linguaggio alla macchina sovrastante $\mathcal{M}_{\mathcal{L}i+1}$
- Indipendenza tra livelli
  - modifiche *interne* alle funzionalità di un livello non hanno influenza sugli altri livelli

Una gerarchia di macchine astratte



Introduzione alla teoria dei linguaggi formali

- Livelli di descrizione di un linguaggio
  - grammatica
    - quali frasi sono corrette?
  - semantica
    - cosa significa una frase corretta?
  - pragmatica
    - come usare una frase corretta e sensata?
  - implementazione (per i linguaggi di programmazione)
    - come eseguire una frase corretta in modo da rispettarne il significato?

Concetto intuitivo di grammatica

- Alfabeto del linguaggio
  - simboli con cui “costruire” le parole del linguaggio
  - es.: alfabeto latino su 22 o 26 lettere
- Lessico
  - parole del linguaggio
  - es.: informatica
- Sintassi
  - determina quali sequenze di parole costituiscono frasi legali (corrette)
- **Le grammatiche sono uno strumento utile per descrivere la sintassi di un linguaggio di programmazione**

Introduzione alla teoria dei linguaggi formali

- Questa prima parte del programma ha l’obiettivo di fornire una base sufficiente alla comprensione delle tecniche di compilazione, oggetto della seconda parte del corso.
- E’ per questa ragione che ci concentreremo quasi esclusivamente su due tipi di grammatiche:
  - regolari
  - libere da contesto

  poiché i linguaggi formali utilizzati in informatica sono per lo più di questi tipi.

Riferimenti

- Maurizio Gabbrielli, Simone Martini. Linguaggi di Programmazione, Principi e paradigmi. McGraw-Hill
  - Capitoli 1-2
- Giovanni Semeraro. Appunti di Teoria dei Linguaggi Formali. Adriatica Editrice, Bari, 1996.
  - Capitolo 1