Guida agli Esami di Linguaggi

VR443470

luglio 2023

Indice

1 Esercizio 1 - Domanda di teoria su Interprete e Compilatore 3

1.1 Interprete 3

1.2 Compilatore 6

2 Esercizio 2 - Induzione 9

2.1 Dimostrare $\forall n \in \mathbb{N}. n + n^2$ è un numero pari 9

2.2 Dimostrare $\sum_{i=1}^{n} \frac{1}{i\,(i+1)} = \frac{n}{n+1}$ 10

2.3 Dimostrare $\sum_{i=0}^{n} i^2 = \frac{n\,(n+1)\,(2n+1)}{6}$ 12

2.4 Dimostrare $\forall n \in \mathbb{N}.\ n > 2$ si ha che $n^2 > 2n + 1$ 14

3 Esercizio 3 - Scoping statico e dinamico 16

3.1 Tipologia codice 1 16

3.2 Tipologia codice 2 16

3.3 Tipologia codice 3 16

3.4 Tipologia codice 4 16

3.5 Tipologia codice 5 16

3.6 Tipologia codice 6 16

3.7 Tipologia codice 7 16

3.8 Tipologia codice 8 16

3.9 Tipologia codice 9 16

4 Esercizio 4 - Scoping (statico/dinamico) e Binding 16

4.1 Regole di scoping e di binding 16

4.2 Codice da inserire in caso di scoping statico/dinamico 16

4.2.1 Tipologia di codice 1 16

4.2.2 Tipologia di codice 2 16

4.2.3 Tipologia di codice 3 16

5 Esercizio 5 - Ricorsione e passaggio di parametri 16

5.1 Ricorsione e ricorsione in coda 16

5.2 Passaggio di parametri: per valore e per riferimento 16

5.2.1 Tipologia di codice 1 16

5.2.2 Tipologia di codice 2 16

5.2.3 Tipologia di codice 3 16

5.2.4 Tipologia di codice 4 16

6 Esercizio 6 - Regole della semantica dinamica 16

6.1 Derivazioni semantica dinamica 16

6.1.1 Tipologia di memoria 1 16

6.1.2 Tipologia di memoria 2 16

6.1.3 Tipologia di memoria 3 16

6.1.4 Vecchi esercizi 16

6.2 Regole della semantica dinamica per il comando condizionale . . 16

6.3 Regole della semantica dinamica per l'assegnamento 16

1 Esercizio 1 - Domanda di teoria su Interprete e Compilatore

1.1 Interprete

In molti esami si presenta la richiesta della definizione di interprete. Nonostante possa essere banale, viene richiesto un "alto" livello di approfondimento dato che vale ben 4 punti all'interno dell'esame. In ogni caso, è possibile affermare che questa domanda sia una delle più gettonate.

Definire intuitivamente e formalmente (mediante definizione semantica) cosa è un interprete. Descrivere la struttura di un interprete e spiegarne il funzionamento.

Risposta

La definizione intuitiva di un interprete è la seguente.

Un **interprete** è un programma $\text{int}^{L_0,L}$ che esegue, sulla macchina astratta per L_0, programmi P^L, i quali sono scritti nel linguaggio di programmazione L, su un input fissato appartenente all'insieme dei dati D (input e output).
Utilizzando parole povere, un interprete non è altro che una "macchina universale" che preso un programma e un suo input, esegue (il programma) sul quel determinato input usando soltanto le funzionalità messe a disposizione dal livello (macchina astratta) sottostante.

Un attimo, ma che cosa si intende per livello? E macchina astratta? Cerchiamo di fare chiarezza.

Dato un linguaggio di programmazione L, la macchina astratta M_L per L è un insieme di strutture dati ed algoritmi che consentono di memorizzare ed eseguire programmi scritti in L. La sua realizzazione può essere fatta in Hardware, Firmware o Software.

Si ma quindi cosa si intende per livello?

Con la scelta della categoria "realizzazione software", vengono utilizzati, per la realizzazione di strutture dati e algoritmi, linguaggi di programmazione ad alto livello poiché essi implementano una struttura suddivisa a livelli di astrazione.
Per concludere, la macchina astratta può essere dunque vista come una stratificazione di livelli dove ciascuno di essi coopera in modo sequenziale, ma allo stesso tempo è indipendente.

Per non lasciare niente al caso, con "linguaggio di programmazione ad alto livello" ci si riferisce a tutti quei linguaggi di programmazione che offrono un livello di astrazione molto alto dei dettagli del funzionamento del calcolatore.

Prima della definizione formale di interprete, si illustrano due notazioni necessarie:

- Con il termine $Prog^L$ ci si riferisce all'insieme dei programmi scritti nel linguaggio di programmazione L;
- Con la dicitura:

  $$\llbracket P^L \rrbracket \,:\, D \to D$$

  Si indica che l'esecuzione del programma scritto nel linguaggio di programmazione L ($\llbracket P^L \rrbracket$) con input in è uguale all'output out. Ovverosia:

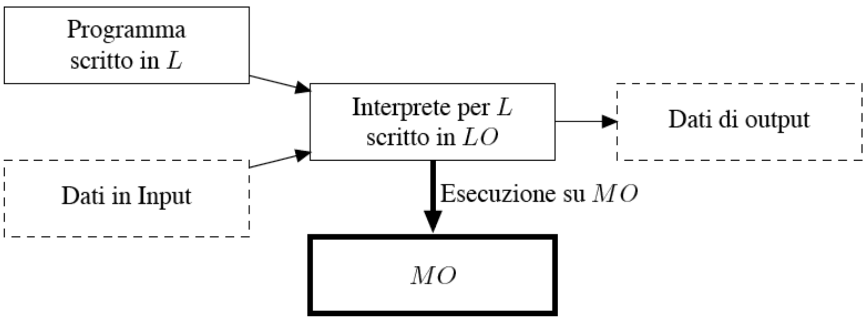
  $$\llbracket P^L \rrbracket (in) = out$$

Un **interprete formalmente** è descrivibile nel seguente modo. Si consideri un interprete da L a L_0: dato $P^L \in Prog^L$ e $in \in D$, un interprete int^{L,L_0} per L su L_0 è un programma tale che:

$$\llbracket int^{L,L_0} \rrbracket \,:\, \left(Prog^L\right) \to D$$

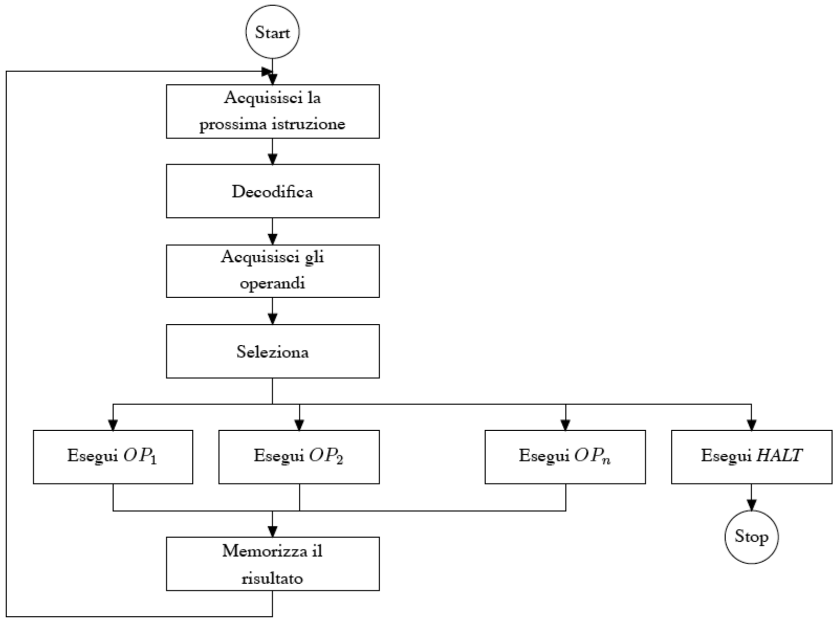
Ne consegue che:

$$\llbracket int^{L,L_0} \rrbracket \,:\, \left(P^L, in\right) = \llbracket P^L \rrbracket (in)$$



Riassunto visivo di quanto appena descritto.

Qua di seguito si presenta il ciclo di esecuzione di un interprete:



1. Il ciclo di esecuzione di un interprete si apre con l'acquisizione di un'istruzione da eseguire;
2. L'operazione del passo precedente viene decodificata;
3. Nel caso in cui l'operazione abbia bisogno di operandi, anch'essi vengono acquisiti dalla memoria;
4. Data l'operazione acquisita al passo 1, viene selezionata una determinata operazione: nel caso di $OP_{...}$ si esegue un'operazione, nel caso di $HALT$ l'esecuzione di un interprete si ferma;
5. Dopo l'operazione eseguita, se vi è un risultato, esso viene salvato. In ogni caso, il ciclo inizia nuovamente la sua esecuzione.

1.2 Compilatore

Non è frequente la richiesta della definizione di compilatore, ma rimane una domanda di teoria che può essere richiesta.

Definire intuitivamente e formalmente (mediante definizione semantica) cosa è un compilatore. Descrivere e spiegare poi la struttura di un compilatore (preferibilmente come flow-chart).

Risposta

La definizione intuitiva di un compilatore è la seguente.

Un **compilatore** è un programma $comp^{L_0,L}$ che traduce, preservando semantica e funzionalità, programmi scritti nel linguaggio di programmazione L in programmi scritti in L_0, e quindi eseguibili direttamente sulla macchina astratta per L_0.

Dato un linguaggio di programmazione L, una macchina astratta M_L per L è un insieme di strutture dati e algoritmi che consentono la memorizzazione e l'esecuzione dei programmi scritti nel linguaggio di programmazione L (P^L).

Prima di dare la definizione formale di interprete, si forniscono alcune notazioni:

- $Prog^L$ è un insieme di programmi scritti nel linguaggio di programmazione L;
- D è l'insieme dei dati (input e output);
- P^L è il programma scritto nel linguaggio di programmazione L;
- $[\![P^L]\!] : D \to D$ rappresenta la semantica di P^L, ovvero l'esecuzione del programma nel linguaggio di programmazione L con input in è uguale all'output out:
$$[\![P^L]\!](in) = (out)$$

La definizione formale di un compilatore è la seguente.

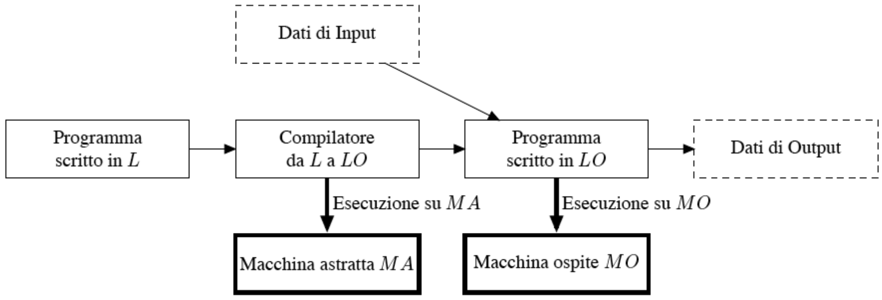
Dato $P^L \in Prog^L$, un **compilatore formalmente** $comp^{L,L_0}$ da L a L_0 è un programma che:

$$[\![comp^{L,L_0}]\!] : Prog^L \to Prog^{L_0}$$

Ovvero:

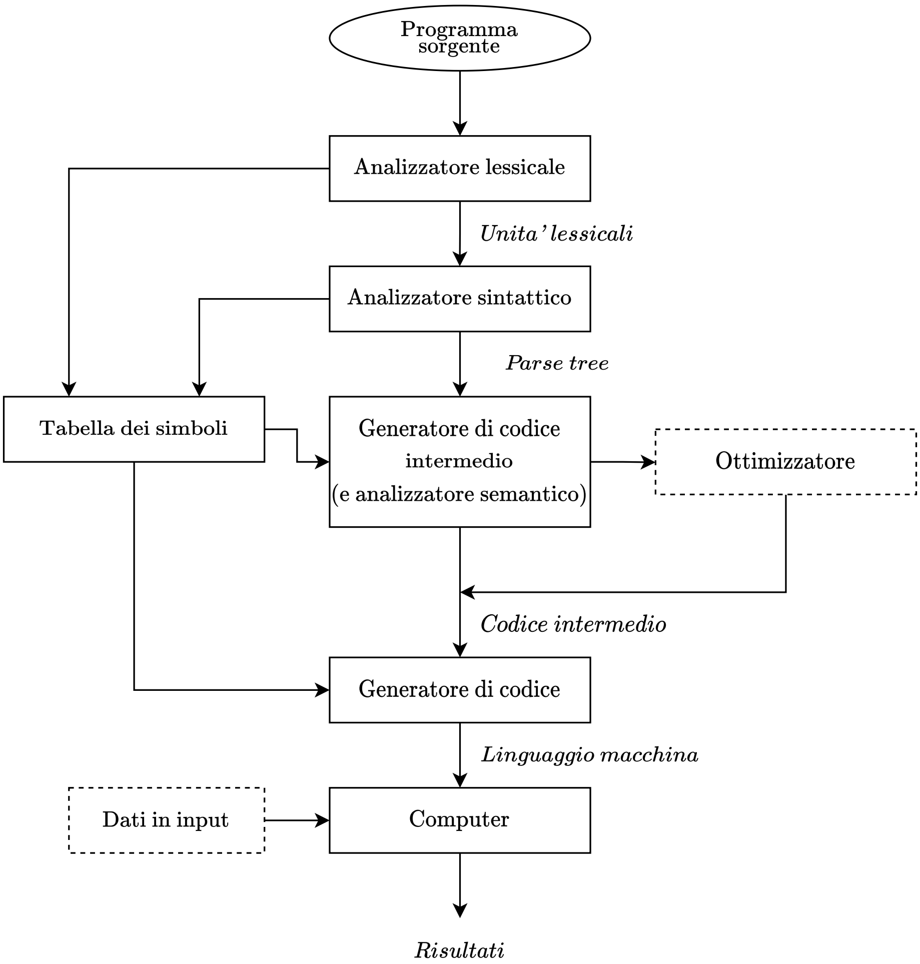
$$[\![comp^{L,L_0}]\!]\left(P^L\right) = \left(P^{L_0}\right) \quad \text{tale che} \quad \forall in \in D.[\![P^{L_0}]\!](in) = [\![P^L]\!](in)$$

In linguaggio non matematico, significa che l'esecuzione del compilatore da L a L_0 insieme (input) ad un determinato programma scritto in un linguaggio di programmazione L è uguale (output) al programma scritto nel linguaggio di programmazione P^{L_0}; tale che per ogni input in appartenente all'insieme dei dati D, l'esecuzione di un programma scritto nel linguaggio di programmazione L_0 con input in è uguale all'esecuzione del programma scritto nel linguaggio di programmazione L con input in.



Flow-chart di quanto detto precedentemente.

Si presenta qua di seguito la **struttura di un compilatore**:



1. Il programma sorgente, da compilare, passa in prima battuta da un analizzatore lessicale. Vengono **convertiti i caratteri del programma sorgente in unità lessicali**. Quest'ultime possono essere *identificatori*, *numeri*, *parole riservate* e formano i linguaggi regolari.

2. Le unità lessicali finiscono in un analizzatore sintattico, il quale crea una albero che rappresenta la sintassi del programma:

   - Le foglie (*token*) vengono lette da sinistra verso destra e costituiscono le frasi ben formate del linguaggio;
   - Dal precedente punto, ne consegue che l'impossibilità della costruzione di un albero è dovuta all'illegalità di quale frase (mal formata, errore di compilazione!);

   Quindi, l'**analizzatore sintattico trasforma unità lessicali in *parse tree* (albero di parse) che rappresentano la struttura sintattica del programma**.

3. Riceve informazioni dall'analizzatore lessicale/sintattico e **memorizza informazioni sui nomi presenti nel programma** (identificatori, chiamate di procedura, ecc.);

4. A questo punto, viene **generato un codice intermedio** che è *indipendente dall'architettura*, e vengono **rilevati eventuali errori semantici** grazie all'*analizzatore semantico*;

5. (Opzionale) Il codice può essere ottimizzato in questa fase;

6. Si conclude la struttura del compilatore con la generazione del codice macchina che, a differenza del codice intermedio, è dipendente dall'architettura.

La parte finale dello schema mostra l'esecuzione del programma compilato su un computer con eventuali dati in input.

2 Esercizio 2 - Induzione

2.1 Dimostrare $\forall n \in \mathbb{N}. n + n^2$ è un numero pari

Dimostrare per induzione che $\forall n \in \mathbb{N}. n + n^2$ è un numero pari.

Risposta

L'**induzione** è una tecnica di dimostrazione che consente di dimostrare la validità di una tesi dalla verifica di due condizioni: la validità della "base induttiva" e la validità del "passo induttivo".

Caso base. Si sceglie come caso base il valore $n = 1$. Si sostituisce:

$$\begin{array}{rcl} n + n^2 & \rightarrow & \text{è pari? } \checkmark \\ & \downarrow & \text{sostituzione di } n = 1 \\ 1 + 1^2 = 2 & \rightarrow & \text{è pari? } \checkmark \end{array}$$

Utilizzando un linguaggio più formale, utile per la dimostrazione, è anche possibile affermare che il modulo di $n + n^2$ diviso 2 è uguale a zero. Ovvero, che dividendo un numero pari per 2 (per definizione), si ottiene il valore zero:

$$\begin{array}{rcl} n + n^2 & = & 0 \pmod 2 \\ & \downarrow & \text{sostituzione di } n = 1 \\ 1 + 1^2 & = & 0 \pmod 2 \end{array}$$

QED

Ipotesi induttiva: si assume che sia vera:

$$n + n^2 = 0 \pmod 2 \quad \forall n \in \mathbb{N}$$

Passo induttivo. Si dimostra che l'ipotesi induttiva implica la validità della proprietà per $n + 1$:

$$\begin{array}{rcl} n + n^2 & = & 0 \pmod 2 \\ & \downarrow & \text{applico passo induttivo} \\ n + 1 + (n+1)^2 & = & 0 \pmod 2 \\ n + 1 + n^2 + 2n + 1 & = & 0 \pmod 2 \\ n^2 + n + 2n + 2 & = & 0 \pmod 2 \\ & \downarrow & \text{applico l'ipotesi induttiva} \\ n^2 + n + 2n + 2 & = & n^2 + n \\ 2n + 2 & = & 0 \pmod 2 \end{array}$$

Per definizione:

- Qualsiasi numero moltiplicato per 2 si ottiene un numero pari;
- Il risultato tra la somma di due numeri pari è ancora un numero pari.

QED

2.2 Dimostrare $\sum_{i=1}^{n} \frac{1}{i\,(i+1)} = \frac{n}{n+1}$

Dimostrare per induzione che $\sum_{i=1}^{n} \frac{1}{i\,(i+1)} = \frac{n}{n+1}$.

Risposta

L'**induzione** è una tecnica di dimostrazione che consente di dimostrare la validità di una tesi dalla verifica di due condizioni: la validità della "base induttiva" e la validità del "passo induttivo".

Caso base. Si sceglie come caso base il valore $n = 1$. Quindi, si va a sostituire:

$$\begin{aligned}
\sum_{i=1}^{n} \frac{1}{i\,(i+1)} &= \frac{n}{n+1} \\
&\downarrow \text{ sostituzione } n = 1 \\
\sum_{i=1}^{1} \frac{1}{i\,(i+1)} &= \frac{1}{1+1} \\
&\downarrow \text{ calcolo della sommatoria} \\
\sum_{i=1}^{1} \frac{1}{1\,(1+1)} &= \frac{1}{1+1} \\
\frac{1}{2} &= \frac{1}{2}
\end{aligned}$$

QED

Ipotesi induttiva: si assume che sia vera:

$$\sum_{i=1}^{n} \frac{1}{i\,(i+1)} = \frac{n}{n+1} \qquad \forall n$$

Passo induttivo. Si dimostra che l'ipotesi induttiva implica la validità della proprietà per $n+1$:

$$\begin{aligned}
\sum_{i=1}^{n} \frac{1}{i\,(i+1)} &= \frac{n}{n+1} \\
&\downarrow \text{ applicazione passo induttivo} \\
\sum_{i=1}^{n+1} \frac{1}{i\,(i+1)} &= \frac{n+1}{(n+1)+1} \\
\sum_{i=1}^{n} \frac{1}{i\,(i+1)} + \frac{1}{(n+1)\cdot((n+1)+1)} &= \frac{n+1}{(n+1)+1}
\end{aligned}$$

$$
\begin{aligned}
& & \downarrow & \quad \text{utilizzo ipotesi induttiva} \\
\frac{n}{n+1} + \frac{1}{(n+1)\cdot((n+1)+1)} &= \frac{n+1}{(n+1)+1} \\
\frac{n}{n+1} + \frac{1}{(n^2+2n+1)+(n+1)} &= \frac{n+1}{(n+1)+1} \\
\frac{n}{n+1} + \frac{1}{(n+1)\,(n+2)} &= \frac{n+1}{(n+1)+1} \\
\frac{n\,(n+2)+1}{(n+1)\,(n+2)} &= \frac{n+1}{(n+1)+1} \\
\frac{n^2+2n+1}{(n+1)\,(n+2)} &= \frac{n+1}{(n+1)+1} \\
\frac{(n+1)^2}{(n+1)\,(n+2)} &= \frac{n+1}{(n+1)+1} \\
\frac{n+1}{n+2} &= \frac{n+1}{n+2}
\end{aligned}
$$

QED

2.3 Dimostrare $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Dimostrare per induzione che $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ **.**

Risposta

L'**induzione** è una tecnica di dimostrazione utilizzata per dimostrare la validità di una tesi grazie alla verifica di due condizioni: la validità della "base induttiva" e la validità del "passo induttivo".

Caso base. Scelgo come caso base il valore $n = 0$. La sua applicazione:

$$\begin{aligned}\sum_{i=0}^{n} i^2 &= \frac{n(n+1)(2n+1)}{6}\\ &\downarrow \text{ sostituisco } n=0\\ \sum_{i=0}^{0} 0^2 &= \frac{0(0+1)(2\cdot 0+1)}{6}\\ 0 &= 0\end{aligned}$$

QED

Ipotesi induttiva: assumo che per $\forall n$ sia sempre vera la seguente proprietà:

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Passo induttivo. Si dimostra che l'ipotesi induttiva implica la veridicità della proprietà per $n+1$:

$$\begin{aligned}\sum_{i=0}^{n} i^2 &= \frac{n(n+1)(2n+1)}{6}\\ &\downarrow \text{ applico il passo induttivo}\\ \sum_{i=0}^{n+1} i^2 &= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6}\\ \sum_{i=0}^{n} i^2 + (n+1)^2 &= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6}\\ &\downarrow \text{ applico l'ipotesi induttiva}\\ \frac{n(n+1)(2n+1)}{6} + (n+1)^2 &= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6}\\ \frac{n(n+1)(2n+1)+6(n+1)^2}{6} &= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6}\end{aligned}$$

$$
\begin{aligned}
2n^3 + n^2 + 2n^2 + n + 6n^2 + 12n + 6 &= \left(n^2 + 3n + 2\right)(2n + 3) \\
2n^3 + 9n^2 + 13n + 6 &= 2n^3 + 9n^2 + 13n + 6
\end{aligned}
$$

QED

2.4 Dimostrare $\forall n \in \mathbb{N}.\, n > 2$ si ha che $n^2 > 2n + 1$

Dimostrare per induzione che $\forall n \in \mathbb{N}.\, n > 2$ ***si ha che*** $n^2 > 2n + 1$ ***.***

Risposta

L'**induzione** è una tecnica di dimostrazione utilizzata per dimostrare la validità di una tesi tramite la verifica di due condizioni: la validità del "passo base" e la validità del "passo induttivo".

Qui.

3 Esercizio 3 - Scoping statico e dinamico

3.1 Tipologia codice 1

3.2 Tipologia codice 2

3.3 Tipologia codice 3

3.4 Tipologia codice 4

3.5 Tipologia codice 5

3.6 Tipologia codice 6

3.7 Tipologia codice 7

3.8 Tipologia codice 8

3.9 Tipologia codice 9

4 Esercizio 4 - Scoping (statico/dinamico) e Binding

4.1 Regole di scoping e di binding

4.2 Codice da inserire in caso di scoping statico/dinamico

4.2.1 Tipologia di codice 1

4.2.2 Tipologia di codice 2

4.2.3 Tipologia di codice 3

5 Esercizio 5 - Ricorsione e passaggio di parametri

5.1 Ricorsione e ricorsione in coda

5.2 Passaggio di parametri: per valore e per riferimento

5.2.1 Tipologia di codice 1

5.2.2 Tipologia di codice 2

5.2.3 Tipologia di codice 3

5.2.4 Tipologia di codice 4

6 Esercizio 6 - Regole della semantica dinamica

6.1 Derivazioni semantica dinamica

6.1.1 Tipologia di memoria 1

6.1.2 Tipologia di memoria 2

6.1.3 Tipologia di memoria 3

6.1.4 Vecchi esercizi

6.2 Regole della semantica dinamica per il comando condizionale

6.3 Regole della semantica dinamica per l'assegnamento