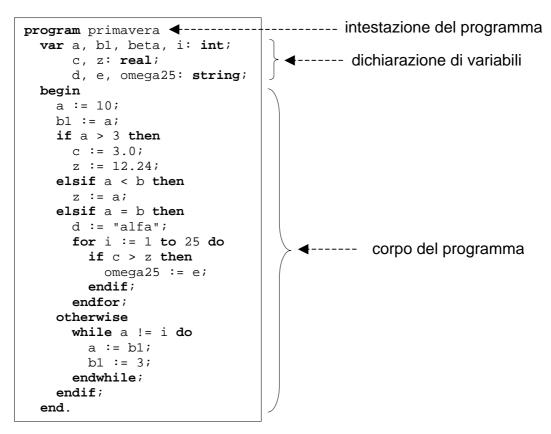
Linguaggi di Programmazione

| Nome e Cognome | |
|-----------------|--|
| Corso di laurea | |
| Telefono | |
| Email | |

1. Specificare la EBNF di un linguaggio imperativo in cui ogni programma è composto da una intestazione, da una sezione (opzionale) di dichiarazione di variabili e da un corpo, come nel seguente esempio:



Le variabili possono essere di tipo int, real o string. Il corpo del programma è costituito da una sequenza non vuota di istruzioni racchiusa tra begin ed end. Ogni istruzione può essere un assegnamento, una istruzione condizionale a più vie, un ciclo a condizione iniziale o un ciclo a conteggio. Possono essere assegnate variabili con altri variabili o valori. Una istruzione condizionale if ... endif coinvolge zero o più rami elsif e, opzionalemte, il ramo otherwise. Una condizione è il confronto (=, !=, >, <, >=, <=) tra una variabile e un valore o un'altra variabile. Ad ogni iterazione del ciclo a conteggio for ... endfor, la variabile di conteggio può incrementare (to) o decrementare (downto). Il corpo dei cicli (for e while), del then e dell'otherwise è costituito da una lista non vuota di istruzioni.

| _ | Cmaaifiaama 1 | a comontico | denotazionale d | | | 0 0 0 0 0 0 | definite | 4.11. | comments | amama ati aa. |
|----|----------------|-------------|-----------------|-----------|-------------|-------------|----------|-------|----------|---------------|
| 7. | -Specificare i | іа ѕешаписа | пепотахтопате п | i iina es | Dressione i | iogica - | аентна | пана | seguenie | grannmanca |
| | | | | | | | | | | |

$$expr \rightarrow true \mid false \mid id \mid expr_1 \text{ and } expr_2 \mid expr_1 \text{ or } expr_2$$

sulla base delle seguenti assunzioni:

- a) id rappresenta il nome di una variabile logica;
- b) la valutazione degli operatori and ed or è in corto circuito;
- c) la valutazione degli operandi dell'operatore **and** è da sinistra a destra;
- d) la valutazione degli operandi dell'operatore **or** è da destra a sinistra;
- e) è disponibile una funzione $\mu(\mathbf{id}, s)$ che restituisce il valore della variabile \mathbf{id} nello stato s;
- f) $\mu(id, s) = \text{UNDEF}$, qualora il valore della variabile id non sia definito;
- g) il linguaggio di specifica denotazionale non dispone di operatori logici.

3. Definire nel linguaggio funzionale *Scheme* la funzione shrink, avente in ingresso una lista L, che computa la lista degli elementi di L che si trovano in posizione dispari, come nei seguenti esempi

| L | (shrink L) | | | | | |
|------------------------------|-------------------|--|--|--|--|--|
| () | () | | | | | |
| (a) | (a) | | | | | |
| (a b) | (a) | | | | | |
| (a b c) | (a c) | | | | | |
| (a (1 2 3) (4 5 (6 7)) 8 ()) | (a (4 5 (6 7))()) | | | | | |

4. Definire nel linguaggio *Haskell* la funzione shrink al punto 3 <u>mediante pattern matching</u>, sulla base del seguente protocollo: shrink :: [a] -> [a].

5. Definire nel linguaggio logico *Prolog* il predicato shrinked (L,S), che risulta vero quando S rappresenta il risultato della funzione shrink (L) specificata al punto 3.

6. Discutere le principali scelte progettuali relative alla definizione di un linguaggio orientato agli oggetti.