RELAZIONE ELABORATO LINGUAGGI (A.A. 2024/2025)

Mattioli Simone (VR486911)

Indice

Introduzione

MyLang è un linguaggio interpretato, progettato interamente da zero, conforme ai 12 requisiti obbligatori definiti dal corso di Linguaggi e Traduttori dell'Università degli Studi di Verona per l'anno accademico 2024/2025.

Il linguaggio MyLang è stato sviluppato utilizzando ANTLR4. L'interprete vero e proprio, invece, è stato implementato in Java, sfruttando il Visitor Pattern per attraversare e valutare l'albero di sintassi astratta (AST) generato da ANTLR4.

Brainfuck può essere eseguito inline in MyLang utilizzando un costrutto specifico (`sly { ... }arnold`).

I requisiti implementati dal linguaggio comprendono:

1. Operazioni aritmetiche di base (`+`, `-`, `*`, `/`, `%`, `^`).
2. Costrutti di non determinismo (`{...} ND [...]`).
3. Cicli condizionali (`while`).
4. Cicli determinati (`for`).
5. Istruzioni condizionali (`if`, `if-else`).
6. Input e output tramite funzioni integrate (`input()`, `print()`).
7. Gestione dinamica delle stringhe e operatore di concatenazione (`++`).
8. Supporto completo ai numeri decimali (`float`).
9. Gestione degli array attraverso liste dinamiche.
10. Variabili dinamiche tramite mappatura in memoria.
11. Definizione e chiamata di funzioni senza parametri.
12. Supporto nativo al linguaggio Brainfuck integrato inline.

Specifiche Funzionali e Implementative

Il linguaggio soddisfa completamente tutte le specifiche richieste:

1. **Operazioni Base:** Supporto alle operazioni aritmetiche (`+`, `-`, `*`, `/`, `%`, `^`) con gestione dinamica dei tipi tramite casting a runtime.
2. **Non Determinismo:** Implementato mediante costrutti `{...} ND [...]` valutati casualmente tramite la classe `Random` di Java.
3. **Ciclo While:** Implementazione standard con valutazione dinamica delle condizioni.
4. **Ciclo For:** Valutazione dinamica di inizializzazione, condizione e aggiornamento.
5. **Blocchi Condizionali:** Supporto completo a istruzioni `if` e `if-else`.
6. **Input e Output:** Gestione integrata delle funzioni built-in `input()` e `print()`.
7. **Gestione Stringhe:** Concatenazione tramite operatore `++` e conversione numerica tramite la funzione built-in `str()`.
8. **Supporto Float:** Parsing e gestione accurata di numeri decimali (`FLOAT`).
9. **Gestione Array:** Implementazione tramite liste dinamiche.
10. **Variabili Dinamiche:** Memorizzate in strutture di tipo `Map<String, Object>`.
11. **Funzioni senza Parametri:** Funzioni con scope locale, registrazione e invocazione tramite una mappa globale.
12. **Brainfuck Integrato:** Supporto completo e isolamento della logica di interpretazione attraverso modalità lexer dedicate (`BF`) e un interprete separato (`BrainfuckInterpreter`).

Architettura del Progetto

Il progetto è organizzato in modo modulare:

- **ANTLR4:** Definisce regole lessicali e sintattiche (`GrammaticaLexer.g4` e `GrammaticaParser.g4`).
- **Visitor Pattern:** La semantica del linguaggio è implementata con `EvalVisitor.java`, che visita l'albero di parsing generato da ANTLR.
- **Gestione Funzioni:** Le definizioni delle funzioni sono gestite centralmente tramite `FunctionRegistry.java`.
- **Interprete Brainfuck:** Il linguaggio Brainfuck viene interpretato separatamente dal resto del linguaggio attraverso `BrainfuckInterpreter.java`, con uno stato gestito tramite la classe `Conf.java`.
- **Normalizzazione Input:** Nel file `Main.java` viene effettuata una normalizzazione minima per evitare problemi legati a caratteri Unicode non previsti.

Dettagli Implementativi

Lexer e Parser

MyLang usa ANTLR4, che permette di definire regole lessicali e sintattiche separatamente, attivando modalità di riconoscimento token specifiche (es. modalità `BF` per Brainfuck).

Gestione Memoria

Ogni ambiente di esecuzione è rappresentato da una `Map<String, Object>`. Durante la chiamata a una funzione, viene creato un nuovo ambiente locale per garantire isolamento e gestione corretta dello scope delle variabili.

Funzioni e Scope

Le funzioni sono definite tramite il costrutto `fun` e registrate in una struttura globale. L'isolamento dello scope viene implementato creando una nuova mappa locale ad ogni invocazione di funzione. Il valore di ritorno è gestito tramite una eccezione interna (`ReturnValue`) per semplificare il flusso di controllo.

Integrazione Brainfuck

L'integrazione Brainfuck è gestita tramite lexer modes specifiche e un interprete separato:

- **Lexer Mode BF:** attivata dal token `sly {`, consente il parsing dei soli simboli validi di Brainfuck.
- **Parser:** riconosce i blocchi Brainfuck (`bfProgram`) e genera un albero specifico.
- **BrainfuckInterpreter:** visita e interpreta l'albero generato, mantenendo uno stato specifico (puntatore dati e memoria).

Test e Validazione

Durante lo sviluppo sono stati eseguiti test approfonditi per ciascun requisito funzionale (array, condizioni, cicli, funzioni, ecc.), garantendo affidabilità e completezza del linguaggio.

Il linguaggio è corredato di demo significative che dimostrano tutte le caratteristiche implementate, incluse la gestione dinamica delle variabili, condizioni complesse, cicli e integrazione di Brainfuck.

Conclusioni

MyLang rappresenta una soluzione completa e robusta che soddisfa pienamente gli obiettivi didattici e tecnici del corso. L'approccio modulare e la chiarezza implementativa lo rendono un progetto ben strutturato, facilmente espandibile e ideale per approfondimenti futuri in ambito accademico e professionale.