

# Compilador para linguagem YA

Mafalda Rosa,40021

Miguel Carvalho, 41136

**Análise Lexical**

Na análise lexical estão definidos todos os tipos que se podem utilizar no trabalho

(int, float, id, string, bool), e palavras reservadas (ex: mod, return, void, if, else,...), e

ainda os literais que se podem utilizar (BOOL\_LIT, INT\_LIT, FLOAT\_LIT,

STRING\_LIT).

Na análise lexical também são ignorados todos os espaços, tabs e new lines, isto

porque por exemplo, um if pode ter o corpo definido na mesma linha, ou ter por

várias, tal como se verifica num dos exemplos de código YA dado pelo professor.

**Análise Sintática e Semântica**

No início do ficheiro são incluídos “includes” onde também se encontra um include

para a apt. Seguidamente são declarados métodos que se usam para a symbol

table (que se encontram definidos no final do ficheiro).

Depois encontra-se uma union que contém todos os tipos utilizados na análise

sintática e para a criação da APT, que se encontram definidos no ficheiro apt.h, e

depois definidos os tokens e types também para realização da análise sintática.

A APT é inicializada em “input:”, sendo que 1º têm de se definir declarações (ids ou

funções ou defines), e nas definições de funções, no corpo, contem statements, que

podem ser declarações, expressões, return, if then else, break, while, next e ainda

utilizar outras funções.

Depois de tudo encontra-se definida a symbol table, que é um array com 256

posições com structs do tipo “valor”, definidas também aqui.

As funções aqui definidas deixa inserir char\*, Ids, Id e fazer look para verificar se já

encontram-se na symbol table.

Em termos do ficheiro de “apt.h”, criamos vários enum para definir os tipos possíveis

para os vários tipos de structs, definimos as ditas structs também e temos as

funções que criam nós para a APT do tipo das structs que definimos anteriormente,

levando como argumentos outros tipos definidos aqui neste ficheiro, ou tipos pré

existentes (char, int, float,..).

A symbol table é utilizada ao longo da criação da APT, tendo os métodos nas

declarações e nas chamadas de funções, assigns, entre outros. Devido à apt ser

criada BOTTOM-UP, há casos em que as declarações dentro de argumentos de

função aparecerem antes da função na symbol table, que pelo tempo apertado e

talvez falta de conhecimento não conseguimos corrigir, mas achamos que está

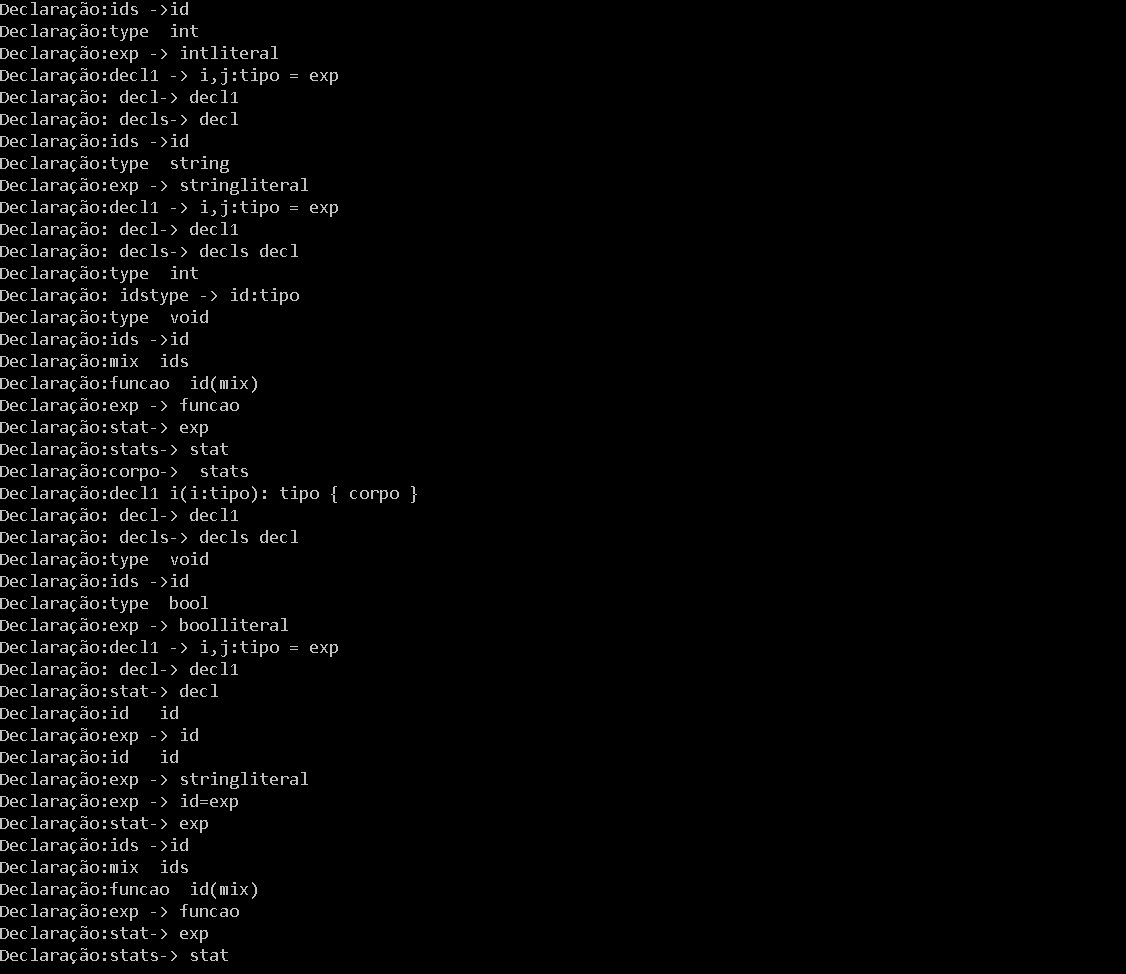
minimamente aceitável.

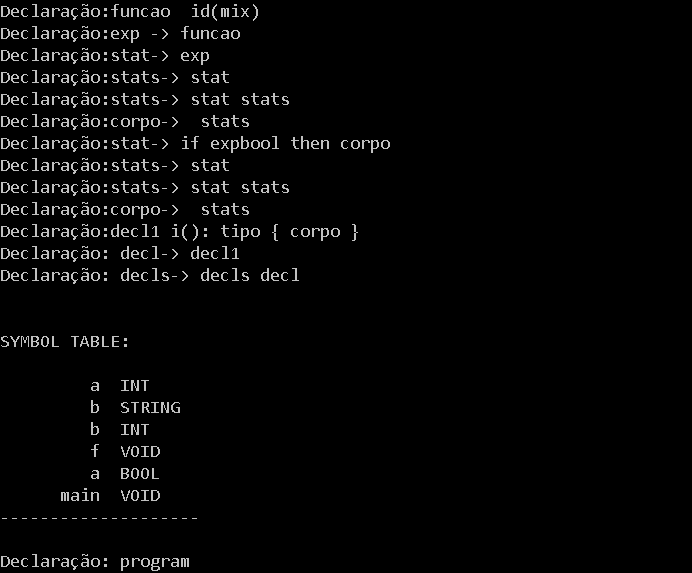
**Comandos:**

make all

./ya < example1.ya

**Exemplo 1:**





Conseguimos assim neste trabalho realizar a análise lexical e sintática

corretamente, sendo que na análise semântica contemos a APT e a symbol table a funcionar minimamente, não tendo o RA para ver alguns conflitos de tipos. Neste trabalho prático não foram assim completados na totalidade os objetivos do enunciado.