**Relatório sobre eFolio Global**

**UC de Compilação na Universidade Aberta**

**Grupo MudMasters Global**

### Constituição do grupo

2000074 Luis Boto - 2000726 José Ciríaco - 2300328 Yara Ibraimo

### Enquadramento

O presente relatório baseia-se no exercício entregue sobre o efólioA e o efólioB, que desenvolveu a gramática MOCC.g4 para a uma linguagem My Own C (MOC) e gerou Árvores Sintáticas Abstratas (AST) com ANTLR.

O efólio Global pretende a implementação de um compilador completo com todas as fases desenvolvidas nos exercícios de avaliação anteriores (efólioA e o efólioB), incluindo as melhorias introduzidas e a geração de um código final.

REVER A PARTIR DAQUI

No efólioA, foi desenvolvida a gramática MOCC.g4 utilizando ANTLR 4 para suportar a sintaxe da linguagem MOC, abrangendo:

* Tipos de dados (int, double, char, void) e vetores (ex.: double v[100]).
* Estruturas de controle (if-else, for, while).
* Expressões aritméticas e lógicas, com precedência definida.
* Chamadas de funções internas (read(), write()) e definidas pelo utilizador (ex.: fact(n)).
* Árvores Sintáticas Abstratas para programas como fatorial.moc, minimo.moc e media.moc, visualizadas no TreeViewer, enfrentando desafios como:
* Ambiguidades Sintáticas: Resolvidas com regras de precedência explícitas (ex.: E → T {+ T} para expressões).
* Validação Semântica Parcial: Declarações duplicadas ou ordem incorreta de protótipos foram tratadas manualmente, sem verificação automática robusta.

As Árvores Sintáticas Abstratas foram cruciais para validar a sintaxe e preparar a transição para o e-fólio B.

Junto com o código podem ser encontrados ficheiros ReadMe com informação adicional tal como os ficheiros teste usados.

Para a realização do efólioB foi necessário resolver as questões pendentes e indicadas como resultado do efólioA.

### Melhorias na Gramática

Para o e-fólio B, aperfeiçoou a gramática MOCC.g4 para:

* Suportar vetores como parâmetros em funções (ex.: double avg(double v[], int size)), ajustando a regra arrayDecl: ID '[' INT? ']'.
* Gerir chamadas recursivas (ex.: fact(k - 1)), com regras claras para listas de argumentos (arguments).
* Separar lexer e parser, eliminando conflitos e melhorando a modularidade. Essas alterações garantiram maior robustez na análise sintática e suporte às necessidades do e-fólio B.

**Análise Semântica**

Implementou-se MOSemanticListener.java para realizar validação semântica, utilizando uma tabela de símbolos baseada em técnicas de compiladores (Aho et al., 2007, Capítulo 7). As principais funcionalidades incluem:

* Deteção de declarações duplicadas (ex.: int x; int x; gera erro).
* Verificação da ordem de protótipos, exigindo que sejam declarados antes de main(void), a tabela de símbolos regista fact como função válida antes de processar main, garantindo consistência semântica.

**Geração de Código Intermédio (TAC)**

Desenvolveu-se TACGenerator.java para percorrer as Árvores Sintáticas Abstratas e traduzir programas MOC em código de três endereços.

Testou-se a geração de TAC com programas como fatorial.moc, media.moc, minimo.moc e erros\_teste.moc. Uma limitação observada é o suporte a recursão: chamadas como fact(k - 1) não são completamente expandidas devido à ausência de uma pilha de ativação no gerador.

**Otimização de Código**

No TACOptimizer.java, implementou-se técnicas de otimização, incluindo:

* Propagação de Constantes: x = 5 \* 1 → x = 5.
* Folding de Expressões: x = 2 + 3 → x = 5.
* Eliminação de Código Morto: Instruções após um return são removidas.

Devido à simplicidade dos programas testados, técnicas mais avançadas, como loop unrolling ou alocação de registradores, não foram aplicadas, mas são reconhecidas como potenciais melhorias.

O efólioB representa um avanço significativo em relação ao efólioA, com a integração de análise semântica, geração de TAC e otimização, fundamentado no uso de Árvores Sintáticas Abstratas robustas. Demonstrando domínio das técnicas de compilação, produzindo um compilador MOCC capaz de processar programas MOC e otimizar código intermédio, apesar de limitações como o suporte a recursão e a validação de tipos. O trabalho atende aos objetivos estabelecidos pela UC, combinando prática teoria.

**Árvore sintática abstrata**

Resultado do ficheiro sem erros “fatorial.moc”

Uma imagem com esboço, desenho, diagrama, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro otimizado “o-fatorial.moc”

Uma imagem com texto, escrita à mão, file, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro sem erros “factorial\_for.moc”

Uma imagem com texto, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro otimizado “o-factorial\_.moc”

Uma imagem com esboço, diagrama, desenho, file

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro sem erros “factorial\_if.moc”

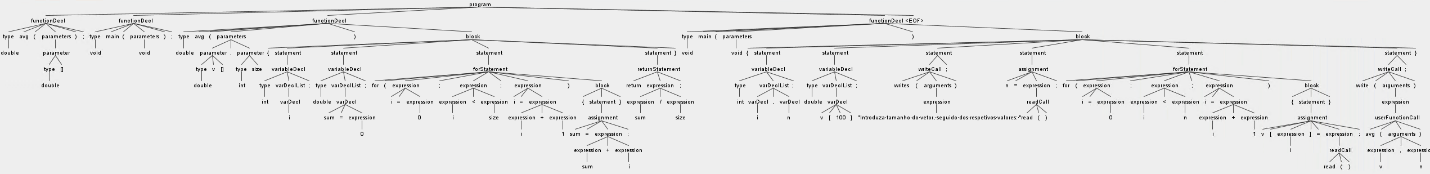
Uma imagem com texto, diagrama, esboço, desenho

Os conteúdos gerados por IA poderão estar incorretos.

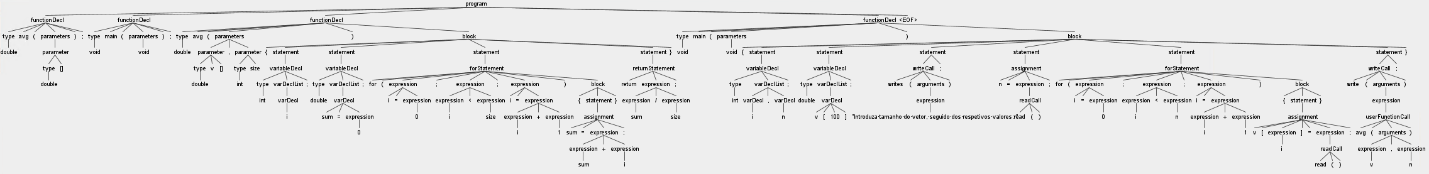
Resultado do ficheiro otimizado “o-factorial\_if.moc”Uma imagem com texto, diagrama, file, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro sem erros “media.moc”



Resultado do ficheiro otimizado “o-media.moc”



Resultado do ficheiro sem erros “minimo.moc”

Uma imagem com diagrama, file, esboço, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro otimizado “o-minimo.moc”

Uma imagem com texto, diagrama, file, branco

Os conteúdos gerados por IA poderão estar incorretos.

Resultado do ficheiro com erros “erros\_teste.moc”

Uma imagem com texto, diagrama, file, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

Tentativa de resolução do ficheiro com erros “o-erros\_teste.moc”

Uma imagem com texto, file, diagrama, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.