

**UNIDADE CURRICULAR:** Compilação

**CÓDIGO:** 21018

**DOCENTE:** Professor Jorge Morais

**A preencher pelo estudante**

**NOMES e N.º DE ESTUDANTE:**

- Gonçalo Caraça, 2000130 (Alt + C ++ Elite)

- Inês Oliveira, 2001090 (The Optimization Optimists)

- Joana Martins, 2003351 (Alt + C ++ Elite)

- João Carvalho, 2103537 (The Optimization Optimists)

- Mário Carvalho, 2000563 (Alt + C ++ Elite)

**CURSO:** Licenciatura em Engenharia Informática

**DATA DE ENTREGA:** 24 de Abril 2023

**TRABALHO / RESOLUÇÃO:**

De acordo com o enunciado, pretende-se, em grupo, construir de raiz um compilador para a linguagem YAIL, apresentando as análises léxica e sintática correspondentes.

Deste modo, e com base nos compiladores “FLEX” e “BISON”, procedeu-se às seguintes etapas:

a)

b)

c)

Durante a sua construção, a equipa deparou-se perante situações, face às quais teve necessidade de resolver, tais como:

- Um analisador sintático (BISON) que funciona-se para C++ adequadamente, e dadas as dificuldades em instalar e usar algo que se ajustasse, alterou-se a linguagem para C, em consenso com todos os elementos do grupo.

Atualmente, encontram-se dois ficheiros adaptados para YAIL, que permitiram realizar testes e enquadrar os resultados esperados, face ao código com erros (YAILcomErros.txt), e face ao código sem erros (YAILsemErros.txt).

A percentagem de testes realizados com sucesso para ambos os ficheiros é de %%%%

Para melhor compreensão do que se realiza durante a leitura do próprio, adaptaram-se as mensagens de alerta de erros, tais como identificação da linha e o tipo de “token” onde se encontra potencialmente um erro. Indicando também a quantidade de erros detetados.

|| + PARA RELATORIO ||  
+ explicar como se considerou o valor interno do vetor ? sem ou com letras?  
+ Parte de sintatica VS parte de Funcionamento (EfolioB)

Corrigir ou confirmar o Inteiro e justificar

RESIZE ABREPARENT IDENT VIRGULA INTEIRO FECHAPARENT

Em que nível está a ser colocadas as Funções\_internas “especiais”? e PQ?  
para serem lidas em todos os níveis, fica em “primeira camada”

**Analisador Léxico (ficheiro flex):**

Inclui *Token’s* que permitem a ligação entre o FLEX e o BISON. Expressa as unidades lexicais definidas pelas expressões regulares e define erro, através da função *yyerror*(), qualquer expressão ou unidade léxica que não corresponde às definidas.

**Analisador Sintático (ficheiro *bison*):**

Inclui função *main*() para lançar programa. Define os *Token’s*, como no ficheiro *Flex*, permitindo a recuperação destes. Define a ação de arranque com utilização do %*start*. Entre %% encontramos a definição da gramática, sem ambiguidades. Na parte final, após %%, o programa arranca através da função *main*() que lê o ficheiro externo e lança as analises léxica e sintática. Todo o código está comentado.

**Opções Tomadas:**

Comentadas na análise do enunciado, reforço contudo a opção pela estrutura para *if*-*else* ser sempre na forma if(condição){bloco instruções} *else* {bloco instruções}, mas o bloco de instruções pode estar vazio. Assim, como para o Output a utilização de sucesso ou erro como solicitado no enunciado, tendo acrescentado para controlo mais alguns prints, no caso dos erros não coloquei o número de linha, mas poderia ser feito com a opção *yylineno*.

# Bibliografia

Aho, A., S. Lam, M., Ravi , S., & D. Ullman, J. (2008). *Compilers: principles, techniques and tools* (2ª ed.). (A. Wesley, Ed., & D. Vieira, Trad.) Universidade Federal de Minas Gerais, Brasil: Pearson.

Reis Santos, P., & Thibault, L. (2015). *Compiladores.* FCA.