



**TRABALHO I**  
**Compiladores**  
**Prof. Lucas Ismaily**

**INFORMAÇÕES IMPORTANTES**

O Trabalho I contém três opções para escolher. As opções são mutuamente exclusivas, isto é, somente é possível escolher **exatamente uma opção**. A data máxima de entrega do trabalho é **28/08/2024**. Porém, recomendo fortemente que entreguem antes, para evitar imprevistos. **Atenção:** findado o prazo de envio, todos os grupos que não enviaram receberão automaticamente nota **zero**. A entrega será **somente** via e-mail (<[E-mail do meu professor](#)>, assunto Trabalho I - Compiladores), numa pasta zipada contendo todos os arquivos, e se preciso, instrução para execução.

**Trabalho pode conter no máximo 5 alunos.** Sejam honestos com vocês e comigo. Qualquer fraude será punida com nota zero para todos os envolvidos.

**PARA AS OPÇÕES 1 E 3 DO TRABALHO CONSIDERE A LINGUAGEM A**

A Linguagem A é definida a partir da Linguagem C, as características da A são:

- Possui apenas os tipos de dados *int* e *string*;
- Não possui laços de repetição e nem condicionais;
- Possui somente os operadores +, -, \*, =, > e <;
- Não possui operadores de bit;
- Não contem funções e nem a possibilidade de iniciar blocos;
- Não contem macros ou importações;
- As demais características são idênticas ao C, inclusive a sintaxe.



### **Opção 1 – Análise Léxica**

1. (2,0 pontos) Crie *Tokens* apropriados e para cada *Token* faça uma Expressão Regular para a Linguagem **A**.
2. (2,0 pontos) Implemente um algoritmo que recebe como entrada todas as Expressões Regulares da Questão anterior e retorna um único Autômato Finito Não-Determinístico (NFA).
3. (3,0 pontos) Implemente um algoritmo que recebe como entrada um Autômato Finito Não-Determinístico (NFA) e retorna um Autômato Finito Determinístico (DFA). A forma de representação dos Autômatos é livre, ou seja, você pode representá-los como matriz, lista, dicionário etc.
4. (3,0 pontos) Utilizando o DFA da Questão 3, implemente um analisador léxico para a Linguagem **A**. Além do código, é preciso entregar um arquivo .txt contendo a lista de *tokens* utilizados e o que eles representam. O arquivo tem o seguinte formato: cada linha contém duas informações separadas por espaço, sendo a primeira posição o *token* e a segunda o que ele representa. Se o *token* representa mais de uma entidade, separe-os por vírgula.

#### **Entrada**

A entrada é composta por um código fonte de um programa qualquer escrito em **A**.

#### **Saída**

Para cada entrada, seu programa deve produzir uma sequência de *Tokens* ou a palavra ERRO, caso a entrada tenha erro léxico.

#### **Exemplo**

##### **Entrada**

```
int a = 0 ;  
in b = 5 + a ;  
string c = "teSte" ;
```

##### **Saída**

```
INT VAR EQ NUM SEMICOLON  
INT VAR EQ NUM ADD VAR SEMICOLON  
STRING VAR EQ CONST SEMICOLON
```



## Opção 2 – Análise Sintática

1. Dada a gramática LR(0) da Figura 1. Você deve Implementar:
  - I. (3,5 pontos) um algoritmo que calcula os conjuntos FISRT e FOLLOW.
  - II. (3,5 pontos) um algoritmo que constrói o Autômato LR(0).
  - III. (3,0 pontos) um algoritmo para o reconhecimento sintático. Isto é, dada uma palavra  $w$ , o seu analisador deve ser capaz de dizer se  $w$  obedece ou não as regras da gramática.

$$\begin{array}{ll} 0 & S' \rightarrow S\$ \\ 1 & S \rightarrow ( L ) \\ 2 & S \rightarrow x \\ 3 & L \rightarrow S \\ 4 & L \rightarrow L , S \end{array}$$

*Figura 1: Gramática LR(0) para ser utilizada.*

### **Entrada**

A entrada é composta por um código fonte de um programa qualquer escrito na gramática escolhida.

### **Saída**

Para cada entrada, seu programa deve produzir uma mensagem de “Sucesso” ou exibir um erro sintático.

### **Exemplo.**

#### **Entrada**

Teste 1. id + id

Teste 2. id \*\* id

#### **Saída**

Teste 1. Sucesso

Teste 2. Erro sintático



### **Opção 3 – Análise Semântica**

1. Implemente um analisador semântico para a Linguagem A. O seu analisador semântico deve verificar:
  - I. (3,5 pontos) se as operações usam tipos compatíveis;
  - II. (3,0 pontos) se as variáveis estão sendo usadas na ordem correta e verificar escopo;
  - III. (3,5 pontos) se as variáveis estão sendo usadas sem serem declaradas.

#### **Entrada**

A entrada é composta por um código fonte de um programa qualquer escrito em A

#### **Saída**

Para cada entrada, seu programa deve produzir uma mensagem de “Sucesso” ou um erro semântico.

#### **Exemplo**

##### **Entrada**

```
int a = 3 ;  
int b = 5 ;  
int c = a + b ;
```

##### **Saída**

Sucesso