Projeto 2: TOC

Construção de Compiladores (INE5426)

Outubro de 2016

1 Equipe do projeto

- **Gerente de Projeto:** Lucas May Petry
- **Projetista de Linguagem:** Luiz Henrique Urias de Sousa
- **Arquiteto do Sistema:** Lucas Machado da Palma
- **Testador:** Luiz Henrique Urias de Sousa

2 Motivação do Projeto

Muitos profissionais de tecnologia já se depararam com a situação pouco confortável de trabalhar com códigos de terceiros, ou mesmo com seus próprios códigos em uma formatação deplorável. Más escolhas de indentação, códigos excessivamente longos, ou até mesmo a sintaxe pouco amigável de algumas linguagens podem colaborar para uma baixa produtividade e o desânimo do desenvolvedor.

A linguagem de programação orientada a objetos TOC tem como objetivo estabelecer um padrão simples e compreensivo de codificação. Os pilares da linguagem são:

- Simplicidade (simplici**T**y): suas palavras reservadas são claras e enxutas colaborando, para a compreensão de seus códigos;
- Organização (**O**rganization): padrões de indentação, nome de funções e classes colaboram para um resultado visualmente elegante;
- Rapidez (qui**C**kness): a linguagem permite uma maior produtividade, proporcionada pela simplicidade e organização do código.

Alavancados por esses princípios, todos os códigos desenvolvidos em TOC podem ser facilmente compreendidos por qualquer desenvolvedor interessado. Consequentemente, a colaboração entre desenvolvedores é facilitada.

3 Pontos a serem implementados no projeto

Característica	**Pontos**
Proposta de nova linguagem	+1
Orientação a objetos	+1
Geração de código intermediário	+2
Tratamento de strings*	+1
Total	5

* Característica adicional da linguagem, não necessariamente será implementada.

4 Características da linguagem

Para alcançar os objetivos mencionados anteriormente, a linguagem TOC utiliza de palavras reservadas de tamanho padrão e regras sintáticas que normalizam a implementação de diferentes estruturas oferecidas em linguagens consolidadas como Java, C++ e Python.

4.1 Básico

- Tipos: inteiro, booleano, ponto flutuante e *string*.
- Condicionais, laços (com iteração sobre vetores) e funções.
- Arrays.
- Comentários.
- Conversão implícita entre tipos.
- Funções primitivas, tais como *len* (obter tamanho) e *print* (imprimir na tela).
- Orientação a objetos, atributos e métodos públicos e privados.
- Operadores relacionais, lógicos e matemáticos.
- Múltiplos escopos (definidos pela indentação).

Para auxiliar o desenvolvimento, durante a compilação quatro tipos de mensagens poderão ser emitidas: erros léxicos, erros sintáticos, erros semânticos e *warnings*. Um *warning* possível seria, por exemplo, alertar o usuário sobre a nomenclatura adequada de funções e classes fortemente recomendadas pela linguagem.

4.2 Adicional

- Concatenação de *strings* e manipulação de *substrings*.
- Diretiva para depuração do código (%expected).

5 Geração de código intermediário

Código intermediário será gerado e interpretado/executado através da biblioteca *LLVM*.

6 Exemplos de código

```
# My first program in TOC

obj Board()
    prv str content

    pub void write(str text)
        my.content = text

    pub str getContent()
        ret my.content

void main()
    Board myBoard()
    myBoard.write("Hello")
    print myBoard.getContent()
```

```
# My second program in TOC

int[] arrayMix(int[] arr)
    for(int i = 0; i < arr.len(); i++)
        for(e in arr)
            if(e > arr[i])
                e = e - random(int)

void main()
    int numbers[3]
    numbers[0] = 43
    numbers[1] = -57
    numbers[2] = 82

    # Does something that I don't know
    numbers = arrayMix(numbers)
```