INE5622 – Introdução a Compiladores

Implementação de Compilador 2 (IC2)

Analise semântica e execução de código

Os trabalhos práticos desta disciplina constituem na elaboração de uma linguagem de programação imperativa e na construção de um interpretador para ela. A execução deles está dividida em duas partes (IC1 e IC2), onde os principais entregáveis, em sequência, são: (IC1) a gramática ANTLR4 usada para gerar o analisador léxico e sintático em Python; e (IC2) o interpretador completo da linguagem com análise léxica, sintática e semântica e execução de código.

Para o contexto dessa entrega, aborda-se a segunda etapa de trabalhos (IC2).

1 Implementação do compilador

Para o desenvolvendo do compilador, os seguintes item **devem ser satisfeitos**:

- A gramática desenvolvida no IC1 deve ser adaptada e usada para desenvolver o analisador semântico da linguagem de programação imperativa que o grupo está desenvolvendo. O interpretador deve reportar erros semânticos com informações suficiente para identificar e corrigir o erro
- O interpretador deve ser capaz de executar qualquer programa escrito na linguagem de programação imperativa que o grupo está desenvolvendo
- Todas as características mínimas apresentadas no enunciado do IC1 devem estar funcionado corretamente na geração de código.

A implementação da característica adicional é desejável, mas não obrigatória.

2 Norma

Para que o trabalho seja avaliado corretamente é **obrigatório que ele sigas as seguintes diretrizes**:

- O interpretador deve ser escrito exclusivamente em Python 3 e deve funcionar em Linux.
- Para testar o compilador, será criado um ambiente virtual Python (virtualenv) onde será instalado suas dependências. Portanto, é mandatório entregar um arquivo `requirements.txt` (exemplo: `https://gitlab.com/evandro-crr/cmm/-/blob/lambda/requirements.txt`).

- O interpretador será executado como um modulo Python (`python -m <modulo>`) e deve ser capaz de receber um código fonte como entrada.

Todas as nomas são satisfeitas pela linguagem de exemplo disponível em `https://gitlab.com/evandro-crr/cmm`.

3 Entrega

O trabalho deve ser feito com a **mesma equipe do IC1** e entregue pelo MOODLE por **apenas um membro da equipe**.

A segunda entrega consiste em um relatório, a implementação completa do interpretador e exemplos de código.

- Entregue em **pdf**, o relatório da entrega IC1 deve ser atualizado, contemplando:
  - A contribuição de cada membro no trabalho.
  - Descrição da implementação do interpretador.
- O interpretador completo escrito em Python e o arquivo `requirements.txt`.
- Exemplos de código correto que abordem **todas as contribuição possíveis da linguagem**; e códigos incorretos que apresentem **todos os erros semânticos**. Adicionalmente, deve ser entregue a implementação da função de Fibonacci na linguagem proposta conforme os códigos de exemplos da Seção 5.

Todos os arquivos devem estar **dentro de um diretório compactado usando zip**.

4 Avaliação

O trabalho valerá nota de 0 a 10, entendendo que o cumprimento de cada item depende da sua descrição no relatório e da implementação no interpretador. O interpretador deve conter todas as **características mínimas enunciadas no IC1** implementadas para poder ser avaliada. Caso essas características forem cumpridas, a pontuação obedecerá a seguinte distribuição:

- Implementação das mensagens de erro semântico: até 3 pontos;
- Implementação da execução de código das características mínimas da linguagem: até 3 pontos;
- Relatório: até 3 pontos;
- Exemplos de código-fonte: até 1 ponto;
- Pontuação extra: implementação da geração de código da características adicional da linguagem: até 3 pontos;

5 Função de Fibonacci

Implementações de referencia da função de Fibonacci (o loop com cin/input e cout/print fazem parte da implementação):

- C++

```
#include <iostream>

int fib(int n) {
    if (n == 1) return 1;
    else if (n == 2) return 1;
    else return fib(n-1)+fib(n-2);
}

int main() {
    for (int i = 0; i < 10; i++) {
        int input;
        std::cin >> input;
        std::cout << fib(input) << std::endl;
    }
    return 0;
}
```

- Python 3

```
def fib(n):
    if n == 1:
        return 1
    elif n == 2:
        return 1
    else:
        return fib(n-1)+fib(n-2)

for _ in range(10):
    print(fib(int(input())))
```

- C−− (linguagem de exemplo da disciplina)

```
def fib(n) {
    if n == 1 return 1;
    if n == 2 return 1;

    return fib(n-1)+fib(n-2);
}
```

```
def main() {
    a = 10;
    while a {
        print fib(input);
        a = a-1;
    }
    return 0;
}
```

6 Links úteis

- Linguagem de exemplo usada na disciplina
  `https://gitlab.com/evandro-crr/cmm`