Exercícios práticos

Construção de Compiladores I

Prof. Rodrigo Ribeiro

Suporte a literate programming para IMP

Introdução

Literate programming é um paradigma para desenvolvimento de programas idealizado por Donald Knuth em 1984. De maneira simples, Knuth argumenta que programas de computador devem ser acompanhados de explicações sobre seu funcionamento em linguagem natural. Ao contrário da abordagem tradicional usada por linguagens atuais em que comentários possuem marcações para diferenciá-los de código, a ideia da **literate programming** é possuir marcações para indicar o código fonte e todo o restante do texto do arquivo é entendido como comentários.

Algumas linguagens, como Haskell e Julia, possuem suporte nativo em seus compiladores para **literate programming** e existem ferramentas, como o **Jypyter** e o **org-mode** do Emacs que permite o uso de **literate programming** para diferentes linguagens de programação.

Neste semestre, vamos adicionar o suporte a **literate programming** no compilador da linguagem Imp, apresentada ao longo do curso BCC328 - Construção de Compiladores I. No que se segue, apresentamos um exemplo do formato de entrada esperado e o que deve ser produzido como resultado pelo compilador protótipo modificado por você.

Formato de entrada esperado

Programas Imp são descritos por arquivos de extensão "**.imp**" e devem conter programas seguindo a sintaxe desta linguagem. Por sua vez, arquivos que usam **literate programming** devem possuir a extensão "**.limp**". Arquivos "**.limp**" podem mesclar código Imp com texto cuja formatação é descrita usando as convenções de Markdown. Linhas contendo código devem iniciar usando o caractere ">" e a ferramenta deve considerar que todo o conteúdo desta linha deve ser processado como um programa Imp tradicional.

Como exemplo, considere o seguinte trecho de código, que mostra o tradicional programa para cálculo do fatorial na linguagem Imp.

```
{
  int n := 0;
  read n;
  int result := 1;
  while 0 < n {
    result := result * n ;
    n := n - 1;
  }
  print result;
}
```

O código anterior não utiliza nenhum recurso de **literate programming**. O trecho a seguir mostra o mesmo exemplo usando o formato "**.limp**".

```
Cálculo de fatorial
===================

Vamos apresentar um programa Imp para o cálculo do fatorial de
um número inteiro.

> {

Primeiro, vamos definir uma variável **n**, de tipo inteiro, e
a inicializamos com o valor **0**.

> int n := 0;

Em seguida, realizamos a leitura do valor do fatorial a ser calculado.

> read n;

O próximo passo é definir uma variável inteira para armazenar o resultado
final. Vamos iniciá-la com o valor **1**.

> int result := 1;

Agora, vamos repetir os próximos passos enquanto o valor **n** for maior que **0**.

> while 0 < n {

Atualizamos o valor da variável **result** para **result * n**.
```

```
>    result := result * n ;

O próximo passo é subtrair **1** do valor atual de **n**.
>    n := n - 1;
>  }

Terminamos por imprimir o resultado.
>  print result;
> }
```

Resultado esperado

Espera-se que a ferramenta processe arquivos "**.imp**" normalmente. A funcionalidade de **literate programming** deve ser executada apenas sobre arquivos "**.limp**" e deve produzir:

1. A interpretação, código SVM ou C a partir do programa de entrada, dependendo do argumentos de linha de comandos fornecidos.
2. Especificação de um fluxograma do código fonte utilizando a linguagem Dot,
   presente na ferramenta Graphviz. Exemplos de como definir fluxogramas utilizando essas ferramentas podem ser encontrados na internet.
3. Um arquivo **.png** contendo uma imagem do fluxograma do código Imp. Pode-se gerar imagens **.png** utilizando o compilador de **Dot**, presente no **Graphviz**.
4. Um arquivo **.md** contendo a documentação, em formato Markdown, presente no arquivo. O arquivo Markdown deverá conter uma referência para um arquivo **.png** do fluxograma.

Evidentemente, para construção desta ferramenta será necessário utilizar quase a totalidade do conteúdo da disciplina BCC328 e, por isso, este será dividido em diferentes etapas.

Entregas

Para facilitar a tarefa de implementação da ferramenta de **literate programming** para Imp, vamos dividí-la em 3 etapas:

1. Construção do analisador léxico: Você deverá construir outro analisador léxico para lidar apenas com arquivos **.limp**.
2. Construção do analisador sintático: Você deverá construir outro analisador sintático para lidar com os diferentes tokens que o formato **literate imp** utiliza.
3. Geração de código: A partir das estruturas de dados produzidas por seu analisador sintático, você deverá gerar código **Dot** e **Markdown** para a produção da documentação.

Cada uma das três etapas deverá ser desenvolvida a partir de um projeto base que conterá a versão completa do compilador / interpretador da linguagem Imp. As datas de entrega de cada uma das etapas serão como se segue:

- Entrega 1.
  - Tema: Análisador léxico.
  - Data limite para entrega: 23/12/2024
- Entrega 2.
  - Tema: Análisador sintático descendente recursivo.
  - Data limite para entrega: 12/01/2025
- Entrega 3.
  - Tema: Analisador sintático LALR e geração de código.
  - Data limite para entrega: 19/03/2025

Todas as entregas deverão ser feitas utilizando repositórios do Github classroom dentro dos prazos estabelecidos.