



Especificação da Linguagem Cordel

1. Introdução

Já que existem muitas linguagens de programação em inglês na computação, neste período vamos implementar uma linguagem de programação em português¹, com uma pequena homenagem à nossa cultura regional.

A linguagem **Cordel** é uma linguagem imperativa, e apresenta as características descritas neste documento.

Cordel, obviamente, é uma linguagem experimental, então esta especificação é passível de adaptações. Em caso de modificações na especificação, versões atualizadas serão postadas via SIGAA (no tópico de aula “Definições do Projeto”) e notificações serão enviadas aos alunos.

As Seções 2, 3, 4 e 5 deste documento apresentam a especificação da linguagem. A Seção 6 contém informações sobre a avaliação e entregas, e o arquivo exemplo.cordel (anexo à aula do SIGAA) contém um exemplo de código `.cordel`.

2. Características e léxico

Regras para identificadores:

- Pode-se utilizar: números, letras maiúsculas, letras minúsculas e *underscore* ('_').
- O primeiro caractere deve ser sempre uma letra.
- Não são permitidos espaços em branco e caracteres especiais (ex.: @, \$, +, -, ^, % etc.).
- Identificadores não podem ser iguais às palavras reservadas ou operadores da linguagem.

Tipos primitivos:

- A linguagem aceita os tipos caractere, booleano, inteiro e real.
- Caracteres são escritos com aspas simples. Exemplo: 'a', '\n'.
- Booleanos podem assumir dois valores: `sim` e `nao`.
- A parte decimal dos números reais deve ser separada por uma vírgula.
- Os valores inteiros são expressos no sistema decimal.

1 Apesar de nossa linguagem ser em português, não utilizaremos acentos por problemas de compatibilidade entre tabelas de codificação em diferentes SOs.

Vetores:

- Um vetor é composto de uma ou mais variáveis com o mesmo tipo primitivo, e expresso com o termo “**ruma de**”.
- O tamanho dos vetores é definido durante sua criação.
- Os índices dos vetores vão de 0 a tam-1.
- Existem apenas vetores unidimensionais.
Exemplo de sintaxe: **ruma de inteiro[3] estoque;**
- Vetores de caracteres podem ter seus valores definidos por cadeias entre aspas duplas (“ e ”).

Blocos

- Blocos são delimitados pelas palavras **inicio** (lembrando: sem acento) e **fim**.

Comentários:

- A linguagem aceita comentários de linha, indicados pelo símbolo # no início da linha
- A linguagem aceita comentários de bloco (possivelmente de múltiplas linhas) delimitados por { e }.
- O funcionamento dos comentários de bloco em Cordel são similares aos da linguagem C.
Exemplo: o que acontece se você compilar /* */ */ em C? E isso: /* /* */? Estudem como um compilador C reconhece o fim de um comentário de bloco.

Estruturas de controle:

- **arrodeie** (similar ao for)
- **enquanto/repita** (similar ao while)
- **se/senao** (sem acento)

Subrotinas:

- Funções com parâmetros e retorno de valor, iniciadas pelo termo “**arrume ... assim:**”. O uso do comando ‘**mande de volta**’ (similar ao return) é obrigatório no corpo da função quando o seu retorno for diferente de vazio (aqui expresso como “nada”). Caso seja vazio, o comando de retorno deve ser omitido.
- Pode haver espaçamento extra entre as palavras “**mande**”, “**de**” e “**volta**”. Ou seja, considerem que pode haver mais que um espaço entre essas palavras.
- Exemplo:

```
arrume inteiro assim: soma(inteiro a | inteiro b)
inicio
    mande de volta a+b;
fim
```

Operadores:

- Operadores aritméticos: +, -, *, /
- Operadores relacionais: >, <, >=, <=, =
- Operadores booleanos: 'não', 'e' e 'ou'.
- A prioridade dos operadores é igual à de C, e pode ser alterada com o uso de parênteses.
- Atribuição de valores é feita com o operador :=
- Comandos são terminados com ; (ponto e vírgula).

3. Sintático

A gramática da linguagem foi escrita em uma versão de E-BNF seguindo as seguintes convenções:

- Variáveis da gramática são escritas em letras minúsculas sem aspas;
- Tokens são escritos entre aspas simples;
- Símbolos escritos em letras maiúsculas representam o lexema de um token do tipo especificado;
- O símbolo | indica produções diferentes de uma mesma variável;
- O operador [] indica uma estrutura sintática opcional;
- O operador { } indica uma estrutura sintática que é repetida zero ou mais vezes.

```
programa : {dec-variavel} {dec-funcao}
dec-variavel : tipo lista-nomes ';'
lista-nomes : ID { ',' ID }
tipo : tipo-base | 'ruma' 'de' tipo-base '[' exp ']'
tipo-base : 'inteiro' | 'caractere' | 'real' | 'booleano'
dec-funcao : 'arrume' tipo-retorno 'assim' ':' ID '(' parametros ')'
bloco
tipo-retorno : tipo | 'nada'
parametros : ε | parametro { '|' parametro }
parametro : tipo ID
bloco : 'inicio' { dec-variavel } { comando } 'fim'
comando :
    'se' '(' exp ')' comando
    | 'se' '(' exp ')' comando 'senao' comando
    | 'enquanto' '(' exp ')' 'repita' comando
    | 'arrodeie' '(' lista-atrib ';' exp ';' lista-atrib ')' comando
    | atrib ';'
    | 'mande' 'de' 'volta' exp ';'
    | bloco
    | chamada ';'

atrib : var ':=' exp
lista-atrib: atrib {, atrib }

var : ID | var '[' exp ']'
exp : INTEIRO | REAL | CARACTERE | BOOLEANO | STRING
    | var
    | '(' exp ')'
    | chamada
    | exp '+' exp
    | exp '-' exp
    | exp '*' exp
    | exp '/' exp
    | exp '=' exp
    | exp '<=' exp
    | exp '>=' exp
    | exp '<' exp
    | exp '>' exp
    | 'não' exp
    | '!' exp
    | exp 'e' exp
    | exp 'ou' exp
```

chamada : ID '(' lista-exp ')'
lista-exp : ϵ | exp { '|' exp }

4. Semântico:

- Nos casos omissos neste documento, a semântica da linguagem segue a semântica de C.
 - Escopo das variáveis: global e local;
 - A execução de um programa consiste na execução de uma função com assinatura:
arrume nada assim: principal(inteiro a | inteiro b)
 - As expressões em se e enquanto devem avaliar um tipo booleano
 - Em qualquer expressão, um caractere tem seu valor automaticamente promovido para inteiro.
 - Em operações entre os tipos inteiro e real, os valores inteiros devem ser convertidos para reais.
 - Existem dois métodos pré-definidos:
 - amostre (): procedimento que recebe como argumento uma expressão e a imprime em tela.
 - espie (): função que retorna o valor inserido pelo usuário no teclado
- Obs: Assumam que os dois métodos são compatíveis com os tipos base e com vetores de caracteres.

O que checar na análise semântica:

- Se entidades definidas pelo usuário (variáveis, vetores e funções) são inseridas na tabela de símbolos - com os atributos necessários - quando são declaradas;
- Se uma entidade foi declarada e está em um escopo válido no momento em que ela é utilizada (regras de escopo são iguais às de C);
- Se entidades foram definidas quando isso se fizer necessário;
- Checar a compatibilidade dos tipos de dados envolvidos nos **comandos**, **expressões**, **atribuições** e **chamadas de função**.

5. Geração de Código

- O código alvo do compilador é C.

6. Desenvolvimento do Trabalho

Trabalhos devem ser desenvolvidos em trio (preferencialmente), dupla ou individualmente. Os grupos devem se cadastrar na planilha:

https://docs.google.com/spreadsheets/d/1tZlEw0Sjf5XdMAxbknPud1k_0Nk6l5BfxhmNfP5gfiM/edit?usp=sharing

Prazo de preenchimento da planilha: 04/03/2022

6.1. Ferramentas

- Implementação com SableCC, linguagem Java.

- IDE Java (recomendação: Eclipse).
- Submissão das etapas do projeto via SIGAA. Será criada uma tarefa para cada etapa.

6.2. Avaliação

- A avaliação será feita com base nas etapas entregues e em arguições feitas com os grupos.
- O valor de cada etapa está definido no plano de curso da disciplina.
- O cumprimento das requisições de formato também será avaliado na nota de cada etapa.

6.3. Entregas

Tarefa 1. Análise Léxica - Parte 1

- **Prazo:** 09/03/2022
- **Atividade:** escrever três códigos em Cordel que, unidos, usem todas as alternativas gramaticais (ou seja, todos os recursos) da linguagem.
- **Formato de entrega:** arquivo comprimido contendo três códigos, onde cada código deve estar escrito em um arquivo de texto simples, com extensão “.cordel”.

Tarefa 2. Análise Léxica - Parte 2

- **Prazo:** 16/03/2022
- **Atividade:** implementar analisador léxico em SableCC, fazendo a impressão dos lexemas e tokens reconhecidos ou imprimindo erro quando o token não for reconhecido.
- **Formato de entrega:** apenas o arquivo .sable deve ser enviado. O nome do arquivo deve ser grupo_X.sable, onde X é o número do grupo (vide planilha de cadastro de grupos). O nome do pacote a ser gerado pelo sablecc deve se chamar cordel (em letras minúsculas).

Tarefa 3. Análise Sintática

- **Prazo:** 06/04/2022
- **Atividade:** implementar analisador sintático em SableCC, com impressão da árvore sintática em caso de sucesso ou impressão dos erros.
- **Formato de entrega:** apenas o arquivo .sable deve ser enviado. O nome do arquivo deve ser grupo_X.sable, onde X é o número do grupo (vide planilha de cadastro de grupos). O nome do pacote a ser gerado pelo sablecc deve se chamar cordel (em letras minúsculas).

Tarefa 4. Sintaxe Abstrata

- **Prazo:** 06/05/2022
- **Atividade:** implementar analisador sintático abstrato em SableCC, com impressão da árvore sintática.
- **Formato de entrega:** apenas o arquivo .sable deve ser enviado. O nome do arquivo deve ser grupo_X.sable, onde X é o número do grupo (vide planilha de cadastro de grupos). O nome do pacote a ser gerado pelo sablecc deve se chamar cordel (em letras minúsculas).

Tarefa 5. Análise Semântica – Parte 1

- **Prazo:** 18/05/2022
- **Atividade:** criação e impressão da tabela de símbolos (a tabela deve ser impressa no console, durante a execução do projeto, sempre que é atualizada).
- **Formato de entrega:** projeto completo, incluindo obrigatoriamente: o arquivo .sable; todas as classes java escritas pelo grupo ou geradas automaticamente; e arquivos .cordel que demonstrem o que foi feito nesta tarefa.
Também é obrigatória a entrega de um pdf contendo uma breve explicação sobre o que foi

implementado nesta etapa e como.

Tarefa 6. Análise Semântica – Parte 2

- **Prazo:** 25/05/2022
- **Atividade:** validar escopo, declaração e definição de identificadores. Implementar verificação de tipos.
- **Formato de entrega:** projeto completo, incluindo obrigatoriamente: o arquivo .sable; todas as classes java escritas pelo grupo ou geradas automaticamente; e arquivos .cordel que demonstrem o que foi feito nesta tarefa.
Também é obrigatória a entrega de um pdf contendo uma breve explicação sobre o que foi implementado nesta etapa e como.

Tarefa 7. Geração de código (extra: 2 pontos):

- **Prazo:** o mesmo da Tarefa 6
- Compilação de código Cordel com geração de código em linguagem alvo (C)

Entregas após o prazo sofrem penalidade de metade da nota da tarefa por dia de atraso.

- Trabalhos entregues com atraso devem ser submetidos na Tarefa ‘Entrega após prazo’, no SIGAA, que ficará aberta durante todo o período. Arquivos enviados por e-mail não serão considerados.

A critério da docente o valor das etapas pode ser modificado, desde que este novo cálculo produza uma nota não menor que a produzida pelo cálculo original.

Bom trabalho!