Construção de Compiladores

Guia da Linguagem MiniC

1. Introdução

Este guia descreve a linguagem de programação MiniC, que será usada como linguagem fonte para o projeto de compilador da disciplina. MiniC é um subconjunto da linguagem C, ou seja, todo programa MiniC é também um programa C, mas há muitas características da linguagem C completa que não aparecem em MiniC. Por exemplo, existem poucos tipos básicos, e não é possível escrever o programa usando mais de um arquivo fonte. Não existem protótipos: cada função usada pela função main deve ser definida antes desta.

Aqui está descrita a estrutura léxica da linguagem. Ao final apresentamos um programa de exemplo simples em MiniC.

2. Estrutura Léxica

A estrutura léxica da linguagem é simples. As classes léxicas são:

- Palavras-chave
- Identificadores
- Literais Inteiro (int)
- Literais de Ponto flutuante (double, float)
- Literais string
- Literais char
- Operadores
- Símbolos de Pontuação
- Comentários

Palavras-chave: Estas são palavras reservadas que não podem ser usadas como identificadores. As palavras-chave do MiniC são:

- **char**
- **else**
- **if**
- **int**
- **float**
- **double**
- **main**
- **printf**
- **printint**
- **printstr**
- **return**
- **while**
- **include**

Identificadores: os identificadores seguem a mesma regra da linguagem C: podem começar com uma letra ou sublinhado (’_’); os demais caracteres podem ser letras, sublinhado ou dígitos.

Literais Inteiros: são literais inteiros, ou seja, cadeias de dígitos numéricos. Por exemplo, 1; 24; 4567, entre outros.

Literais de Ponto flutuante: são literais reais, como por exemplo, 2.55, 45.678, 0.75, entre outros.

Literais caracteres: seguem a regra da linguagem C, sendo compostos por um caractere envolto em aspas simples. O caractere pode ser uma das sequências de escape ‘\r’, ’\n’ ou ’\t’. Para representar uma barra invertida como caractere, deve-se usar ‘\\’.

Literais String: mesma regra da linguagem C: começam e terminam com um caractere de aspas duplas, e podem conter as sequências de escape \r, \n e \t. Para incluir uma barra invertida (\) na string, deve-se usar \\. Uma string representa um array de caracteres. Por exemplo: **“\nHello World\n”**.

Operadores: os operadores binários em MiniC são as quatro operações aritméticas (+, -, * e /), os operadores de comparação (==, !=, <, >, <= e >=) há também o operador de atribuição (=). O único operador unário é o de negação lógica (!).

Símbolos de Pontuação: os símbolos de pontuação na linguagem MiniC são os tipos básicos encontrados na linguagem C, sendo eles **: ; , ( ) { } . #**

Comentários: seguem as mesmas regras da linguagem C: comentários até o final da linha começando com // ou comentários multilinha começando com /* e terminando com */. Comentários multilinha não podem ser aninhados.

3. Gramática

Um programa MiniC sempre deve estar contido em apenas um arquivo. É preciso ter pelo menos uma função, a função principal, chamada de main. Antes da função principal pode ser definido um número qualquer de outras funções, que poderão ser chamadas no **main** ou em outras funções. As estruturas de controle são apenas if para comandos condicionais e **while** para loops. Os tipos básicos são: inteiros (**int**), double (**double**), float (**float**) e caracteres (**char**). O único tipo composto é o **array**, que pode ter componentes **int, float, double** ou **char**.

A gramática usa a notação N∗, onde N é um não-terminal, para denotar nenhuma, uma ou várias ocorrências de N. Os terminais em negrito são palavras chave. Para as expressões aritméticas, assume-se que os operadores têm precedência e associatividade padrão. Os operadores aritméticos têm maior precedência que os de comparação, e estes têm precedência maior do que os operadores lógicos. As funções **printint** e **printstr** imprimem um inteiro e uma string, respectivamente; são fáceis de escrever em C padrão usando **printf**. A gramática completa para a linguagem é apresentada a seguir.

Programa → DeclInclude* FuncaoPrincipal DeclFuncoes*

DeclInclude → #**include**<ID.h>

DeclFuncoes → Tipo ID (ListaArgumentos) {Comando* **return** Expressao;}

FuncaoPrincipal → **int main**() { Comando* **return** Literal Inteiro; }

Tipo → **int** | **float** | **double** | **char**

Comando → {Comando*} |

if (ExpRel) Comando **else** Comando |

if (ExpRel) Comando |

while (ExpRel) Comando |

printf (Expressao); |

printint (Literal Inteiro); |

printstr (Literal String); |

Tipo ID (, ID)*; |

Tipo ID = Expressao **(**, ID = Expressao**)***; |

ID = Expressao;

ExpRel → ExpRelAux OpRelacional ExpRelAux |

!(ExpRel)

ExpRelAux → ID |

Literal Inteiro |

Literal Ponto Flutuante

OpRelacional → > | < | == | >= | <= | !=

ExprAritmetica → ExprAritmetica + T |

ExprAritmetica - T |

T

T → T * F |

T / F |

F

F → (ExprAritmetica) |

ID |

Literal Inteiro |

Literal Ponto Flutuante |

ID (Parametros*)

Expressao → ExprAritmetica |

Literal Char |

Literal String

Parametros → Expressao **(**, Expressao**)***

4. Ações Semânticas

O objetivo desta última parte do projeto é implementar todas as ações semânticas que serão executadas à medida que a análise sintática for sendo realizada. O objetivo das ações semânticas é criar uma representação do programa lido na memória usando classes e objetos em Java. Observe o exemplo a seguir:

```
/* Simbolos Não Terminais */

non terminal ProgramaMiniC ProgramaMiniC;

/* Produções */

start with ProgramaMiniC;

ProgramaMiniC ::= DeclIncludeOpt:includes FuncaoMain:funcaoMain
DeclFuncoesOpt:declFuncoes {: RESULT = new ProgramaMiniC( includes,
funcaoMain, declFuncoes ); :};
```

Um programa em MiniC é formado por zero ou várias instruções include, seguido da declaração da função Main e por fim por zero ou mais declarações de funções. Para representar um ProgramaMiniC em memória a ação semântica desenvolvida cria um objeto da classe ProgramaMiniC que recebe por parâmetros uma lista de declarações include, a função Main e uma lista de declarações de funções.

Esse procedimento deve ser repetido para todas as produções de forma que ao executar o código da classe CompiladorMiniC, caso o programa de entrada esteja correto, o compilador deve exibir na tela o mesmo código de entrada, com exceção dos comentários, usando o trecho de código a seguir:

```
Symbol symbol = parser.parse();

ProgramaMiniC programaMiniC = (ProgramaMiniC) symbol.value;
```

System.out.println(programaMiniC);

5. Exemplos

A Figura a seguir apresenta um exemplo simples de programa que calcula o fatorial do número escrito na linguagem MiniC.

```
#include <stdio.h>
#include <stdlib.h>

// Programa Teste Projeto Compiladores

int main() {

  int a = 20;
  float f = 2.5;
  char a = 'A';

  printf("\n Hello World \n" );

  return 1;

}

int fatorial(int n) {

  int res;

  if (n < 1)
    res = 1;
  else
    res = n * fatorial(n - 1);

  return res;

}
```

4. Informações Importantes

Data Entrega: 09/12/2018 (via Unipê virtual) às 23:55.

Equipe: Grupo de no máximo 3 alunos.

Ferramentas: JavaCUP e JFlex.

Observação. Ao final do trabalho, cada aluno deverá fazer o **upload do projeto completo**, incluindo:

- **a)** Um arquivo contendo o nome e a matricula de cada aluno do grupo.
- **b)** Os arquivos de especificação (.flex) e (.cup).
- **c)** As classes do Analisador Léxico e Sintático gerados.
- **d)** As classes que geram o Analisador Léxico e o Analisador Sintático a partir do arquivo de especificação.
- **e)** A classe principal que utiliza os Analisares léxico e sintático gerados passando m arquivo de teste como entrada.
- **f)** Todas as classes que representam os elementos que precisam ser reconhecidos na linguagem.

Nota: 3,0.

5. Notas Bibliográficas

A linguagem MiniC foi inicialmente baseada na linguagem MiniJava, apresentada no livro de Appel sobre compiladores em Java [1].

Referências

[1] Appel, Andrew W. e Palsberg, Jens, Modern Compiler Implementation in Java, 2nd ed., (Cambridge: Cambridge University Press, 2002).