Atividade de Compiladores

Parte A: Analisador Léxico para a Linguagem

Linguagem: L = {w | w é uma expressão matemática contendo no máximo 2 operadores sobre números naturais}

Alfabeto: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, (,)}

Código Flex para o Analisador Léxico:

```
%{
   #include <stdio.h>
   #include <stdlib.h>
%}

digit       [0-9]
operator    [+-]
whitespace  [ \t]
parentheses [\(\)]
endl        [\n]

%%

{digit}+                { fprintf(yyout, "NUM -> %s\n", yytext); }
{operator}              { fprintf(yyout, "OP -> %s\n", yytext); }
{parentheses}           { fprintf(yyout, "PARENTESES -> %s\n", yytext); }
{whitespace}+           { /* Ignora espaços em branco */ }
{endl}                  { fprintf(yyout, "\n"); }
.                       { fprintf(yyout, "Caractere inválido: %s\n", yytext); exit(1); }

%%

int main(int argc, char **argv)
{
   if (argc != 3) {
      fprintf(stderr, "Uso: %s <arquivo de entrada> <arquivo de saída>\n", argv[0]);
      return 1;
   }

   FILE *input = fopen(argv[1], "r");
   if (!input)
   {
      perror("Erro ao abrir o arquivo de entrada");
      return 1;
   }

   FILE *output = fopen(argv[2], "w");
```

```
    if (!output)
    {
        perror("Erro ao abrir o arquivo de saída");
        fclose(input);
        return 1;
    }

    yyin = input;
    yyout = output;

    yylex();

    fclose(input);
    fclose(output);

    return 0;
}
```

Parte B: Gramática para L:

S -> E
E -> T | E + T | E - T
T -> F | T + F | T - F
F -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Essa gramática define expressões que consistem em termos (`T`) que podem ser somados ou subtraídos, onde cada termo pode ser um único número (`F`) ou uma expressão composta.

Parte C: Verificação de Ambiguidade da Gramática

A gramática fornecida é ambígua porque existem múltiplas maneiras de derivar a mesma expressão, o que leva a diferentes árvores de derivação. Por exemplo, a expressão "1 + 2 - 3" pode ser interpretada como (1 + 2) - 3 ou como 1 + (2 - 3), o que resulta em diferentes resultados.

Problemas com Gramáticas Ambíguas:

- Dificuldade na Análise Sintática: Um analisador sintático pode ter dificuldades em decidir qual derivação seguir.
- Interpretação Incorreta: A ambiguidade pode resultar em cálculos incorretos devido à má interpretação da ordem das operações.
- Complicações na Implementação: Implementar um parser para uma gramática ambígua é mais complexo e propenso a erros.

Parte D: Analisador Léxico e Gramática com Parênteses

Código Flex Atualizado:

```
%{
   #include <stdio.h>
   #include <stdlib.h>
%}

digit       [0-9]
operator    [+-]
whitespace  [ \t\n]
parentheses [\(\)]
endl        [\n]

%%

{digit}+                { fprintf(yyout, "NUM -> %s\n", yytext); }
{operator}              { fprintf(yyout, "OP -> %s\n", yytext); }
{parentheses}           { fprintf(yyout, "PARENTESES -> %s\n", yytext); }
{whitespace}+           { /* Ignora espaços em branco */ }
{endl}                  { fprintf(yyout, "\n"); }
.                       { fprintf(yyout, "Caractere inválido: %s\n", yytext); exit(1); }

%%

int main(int argc, char **argv)
{
   if (argc != 3) {
      fprintf(stderr, "Uso: %s <arquivo de entrada> <arquivo de saída>\n", argv[0]);
      return 1;
   }

   FILE *input = fopen(argv[1], "r");
   if (!input)
   {
      perror("Erro ao abrir o arquivo de entrada");
      return 1;
   }

   FILE *output = fopen(argv[2], "w");
   if (!output)
   {
      perror("Erro ao abrir o arquivo de saída");
      fclose(input);
      return 1;
   }

   yyin = input;
   yyout = output;
```

```
    yylex();

    fclose(input);
    fclose(output);

    return 0;
}
```

Gramática Atualizada com Parênteses:

S -> E
E -> E + T | E - T | T
T -> (E) | F
F -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Essa gramática inclui parênteses para permitir o agrupamento de expressões, o que resolve parte da ambiguidade ao impor uma ordem de operações clara.

Parte E: Os Parênteses Fazem Diferença?

Sim, os parênteses fazem diferença na gramática. Eles são usados para alterar a ordem natural das operações. Sem parênteses, as operações são realizadas em uma ordem específica (geralmente da esquerda para a direita). No entanto, os parênteses podem alterar essa ordem, garantindo que as operações dentro dos parênteses sejam realizadas primeiro.

Explicação:

- Sem Parênteses: A expressão 1 + 2 - 3 pode ser ambígua e pode ser interpretada de diferentes maneiras.
- Com Parênteses: A expressão (1 + 2) - 3 ou 1 + (2 - 3) não deixa espaço para ambiguidade, pois a ordem das operações é explicitamente definida.

A inclusão de parênteses na gramática ajuda a remover a ambiguidade, pois força a avaliação das expressões na ordem desejada.