Relatório Final - Tradutor Matemágica

Introdução

Este trabalho apresenta a implementação de um tradutor para a linguagem "Matemágica", uma linguagem fictícia que permite operações aritméticas básicas, comandos condicionais e loops. O tradutor converte o código Matemágica para código C utilizando as bibliotecas `ply.lex` e `ply.yacc` para análise léxica e sintática.

Implementação

O que foi implementado

A implementação consiste em um analisador léxico e sintático, além de um gerador de código que traduz o código Matemágica para C. A linguagem Matemágica possui os seguintes comandos e funcionalidades:

- **Declaração e inicialização de variáveis**: Exemplo, `FACA x SER 5.` traduz para `int x = 5;` em C.
- **Operações aritméticas básicas**: `SOME`, `MULTIPLIQUE`, que traduzem para somas e multiplicações em C.
- **Comando condicional `SE-ENTAO-SENAO`**: Executa uma verificação e toma uma ação conforme o valor de uma variável.
- **Loop `REPITA N VEZES`**: Traduzido para um loop `for` em C.
- **Comando de impressão `MOSTRE`**: Traduzido para `printf` em C, permitindo exibir variáveis e números.

Como foi implementado

A implementação foi realizada em Python com o uso das bibliotecas `ply.lex` e `ply.yacc` para construção do lexer e parser.

1. **Lexer**: Identifica tokens da linguagem Matemágica, como `FACA`, `SER`, `SOME`, `MULTIPLIQUE`, `ID` (identificadores) e `NUMBER` (números).
2. **Parser**: Define a gramática Matemágica, onde cada regra da gramática traduz um comando para código C correspondente.
3. **Gerador de Código**: Constrói o código C final, armazenado na variável `generated_code`, que é atualizado conforme o parser interpreta comandos Matemágica.

O que funciona

- Todos os comandos Matemágica especificados no enunciado foram implementados e funcionam corretamente.
- O código gerado em C é válido e pode ser compilado e executado para realizar as operações Matemágica.

O que não funciona

- Não foram encontradas limitações significativas na implementação.

Testes Realizados

Foram realizados 8 testes para verificar o funcionamento completo do tradutor Matemágica para C. Cada teste foi executado e comparado com a saída esperada.

1. Teste de Atribuição e Impressão

Código Matemágica:

```
```

```
FACA x SER 10.
MOSTRE x.
```

Código C Esperado:

```c
int x = 10;
printf("x = %d\n", x);
```

2. Teste de Atribuição com Condicional

Código Matemágica:

```
FACA y SER 20.
SE y ENTAO
MOSTRE y.
SENAO
MOSTRE 0.
FIM
```

Código C Esperado:

```c
int y = 20;
if (y != 0) {
    printf("y = %d\n", y);
} else {
    printf("%d\n", 0);
}
```

3. Teste de Soma entre Variáveis

Código Matemágica:

```
FACA a SER 5.
FACA b SER 10.
SOME a COM b.
MOSTRE a.
```

Código C Esperado:

```c
int a = 5;
int b = 10;
a = a + b;
printf("a = %d\n", a);
```

4. Teste de Multiplicação entre Variáveis

Código Matemágica:

```
FACA m SER 3.
FACA n SER 4.
MULTIPLIQUE m POR n.
MOSTRE m.
```

Código C Esperado:

```c
int m = 3;
int n = 4;
m = m * n;
printf("m = %d\n", m);
```

5. Teste de Loop `REPITA` com Operações

Código Matemágica:

```
FACA contador SER 1.
REPITA 3 VEZES:
MULTIPLIQUE contador POR 2.
FIM
MOSTRE contador.
```

Código C Esperado:

```c
int contador = 1;
for (int _ = 0; _ < 3; _++) {
    contador = contador * 2;
}
printf("contador = %d\n", contador);
```

6. Teste de Condicional e Loop

Código Matemágica:

```
FACA soma SER 0.
REPITA 5 VEZES:
SOME soma COM 1.
FIM
SE soma ENTAO
MOSTRE soma.
SENAO
MOSTRE 0.
FIM
```

Código C Esperado:

```c
int soma = 0;
for (int _ = 0; _ < 5; _++) {
    soma = soma + 1;
}
if (soma != 0) {
    printf("soma = %d\n", soma);
} else {
    printf("%d\n", 0);
}
```

7. Teste de Atribuição, Condicional e Impressão

Código Matemágica:

```
FACA z SER 5.
SE z ENTAO
MOSTRE z.
SENAO
MOSTRE 0.
FIM
```

Código C Esperado:

```c
int z = 5;
if (z != 0) {
    printf("z = %d\n", z);
} else {
    printf("%d\n", 0);
}
```

8. Teste Completo com Várias Funcionalidades

Código Matemágica:

```
FACA contador SER 1.
FACA soma SER 0.
REPITA 5 VEZES:
SOME soma COM contador.
MULTIPLIQUE contador POR 2.
FIM
SE soma ENTAO
MOSTRE soma.
SENAO
MOSTRE 0.
FIM
```

Código C Esperado:

```c
int contador = 1;
int soma = 0;
for (int _ = 0; _ < 5; _++) {
    soma = soma + contador;
    contador = contador * 2;
}
if (soma != 0) {
    printf("soma = %d\n", soma);
} else {
    printf("%d\n", 0);
}
```

Gramática Final

Regras de Produção

```
programa : cmds

cmds : cmd cmds
     | cmd

cmd : FACA ID SER NUMBER DOT
    | MOSTRE ID DOT
    | MOSTRE NUMBER DOT
    | SOME ID COM ID DOT
    | SOME ID COM NUMBER DOT
    | MULTIPLIQUE ID POR ID DOT
    | MULTIPLIQUE ID POR NUMBER DOT
    | REPITA NUMBER VEZES COLON cmds FIM
    | SE ID ENTAO cmd FIM
    | SE ID ENTAO cmd SENAO cmd FIM
```

Explicação das Regras

- **programa : cmds**
  - Define um programa como uma sequência de comandos (cmds).
- **cmds : cmd cmds | cmd**
  - Define uma lista de comandos, que pode conter um ou mais comandos.
- **cmd : FACA ID SER NUMBER DOT**
  - Atribuição de um valor a uma variável (`int ID = NUMBER;`).
- **cmd : MOSTRE ID DOT ou MOSTRE NUMBER DOT**
  - Comando de impressão (`printf`).
- **cmd : SOME ID COM ID DOT ou SOME ID COM NUMBER DOT**
  - Comando de soma entre variáveis ou entre uma variável e um número.
- **cmd : MULTIPLIQUE ID POR ID DOT ou MULTIPLIQUE ID POR NUMBER DOT**

    - Comando de multiplicação entre variáveis ou entre uma variável e um número.

- **cmd : REPITA NUMBER VEZES COLON cmds FIM**
    - Loop REPITA N VEZES, traduzido para um for em C.

- **cmd : SE ID ENTAO cmd FIM ou SE ID ENTAO cmd SENAO cmd FIM**
    - Comando condicional SE-ENTAO-SENAO, que verifica o valor de uma variável.

Regras Adicionadas

As seguintes regras foram adicionadas para suportar funcionalidades específicas:

- **MULTIPLIQUE ID POR NUMBER DOT**
    - Adicionada para permitir multiplicação entre uma variável e um número.
- **SE ID ENTAO cmd SENAO cmd FIM**
    - Adicionada para suportar a estrutura condicional SENAO além do SE-ENTAO.

Conclusão

O projeto foi concluído com sucesso, permitindo que a linguagem Matemágica seja traduzida para código C. Todos os comandos foram testados e estão funcionando conforme esperado, produzindo saídas corretas em C para os casos de teste especificados.

Os testes realizados demonstram que o tradutor é capaz de lidar com todos os casos de uso previstos, desde operações simples até combinações complexas de estruturas de controle. A gramática implementada mostrou-se suficiente para expressar todos os comandos necessários da linguagem Matemágica.