Disciplina: Construção de Compiladores

Professor: José Carlos Toniazzo

Alunos: Silvana Trindade e Maurício André Cinelli

Analisador Léxico para a Linguagem Jusm

Guia da linguagem Jusm

A linguagem jusm é baseada em pseudocódigo, o que a torna de simples compreensão.

Inserir comentários

```
:: comentário inserido com sucesso
```

Escopo da main

```
main begin
        ::corpo
end
```

Declaração e atribuição de variável do tipo inteiro

```
int x
x = 1
```

Declaração e atribuição de variável do tipo ponto flutuante

```
float y
y = 2,5 * x
```

Declaração e atribuição de variável do tipo string

```
string juice
juice = “apple”
```

Imprimir no terminal

```
print “ ” juice
```

Entrada pelo terminal

```
scan juice
```

Declaração de if

```
if x < 3 begin
```

```
        ::corpo
    end
Declaração de while
    while x > 10 begin
        ::corpo
    end
Declaração de for
    for x  in 1 to y begin
        ::corpo
    end
```

Exemplo de algoritmo :

```
::Exemplo 1
::Algoritmo que calcula o fatorial de um número x utilizando for
main begin
    int x
    int fatorial = 1 ::inicializa fatorial com um pois 0! = 1
    int i = 1
    scan x ::leitura do número que deseja obter o fatorial

    for i in 2 to x begin
        fatorial = fatorial * i
    end
    print " " fatorial
end

::Exemplo 2
::Algoritmo que calcula o fatorial de um número x utilizando while
main begin
```

```
        int x
        int fatorial = 1 ::inicializa fatorial com um pois 0! = 1
        int i = 1::iterador
        scan x ::leitura do número que deseja obter o fatorial
        while i < x begin
                fatorial = fatorial * i
                i = i+1
        end
        print " "fatorial
end
```

Conjunto de Palavras Reservadas

1. main
2. begin
3. end
4. int
5. float
6. string
7. if
8. while
9. for
10. print
11. scan
12. to
13. in

Operadores

1. +
2. -
3. *
4. /
5. %
6. <
7. >
8. <=
9. >=
10. ==
11. !=
12. !
13. =
14. (
15.)
16. and
17. or

Declaração de variáveis

- ➔ Não começar com caractere especial, exceto underscore (_)
- ➔ Não começar com número
- ➔ Não começar com operadores
- ➔ Não ser igual a palavras reservadas
- ➔ Não ter acento

Implementação do analisador Léxico

Analisador léxico desenvolvido em linguagem C, compilada em sistema operacional Linux.

Buffer

Foi utilizado buffer alternado, cada um com 8 bytes.

Expressões regulares

As expressões regulares utilizadas são simples, por exemplo para números com ponto flutuante:

(0|1|2|3|4|5|6|7|8|9)+,(0|1|2|3|4|5|6|7|8|9)*

as demais podem ser visualizadas a partir da linha 28 no arquivo *analisador.*c .

Algoritmo de Thompson

O algoritmo de Thompson transforma expressões regulares em autômato finito não determinístico. Posteriormente converte o autômato em uma máquina determinística, com a finalidade de reconhecer uma gramática, neste caso a da linguagem jusm.

Tratamento de erros

O tratamento de erros utilizado foi o modo pânico, pois o grupo observou que linguagens tradicionais como C e C++ utilizam do mesmo. Os erros léxicos verificados foram: simbolos não pertencente a um conjunto, variável utilizada mas não declarada e, tratamento para string não fechada.

Exemplos de erros:

#$%#%@%T	:: lixo na linha
float a = 2,@	::ponto flutuante mal formado
string name = “Jude	::não tem ‘ ” ’
int a = (b+c) / (4+k)	:: onde b por exemplo não foi declarado

Tabela de Tokens

Para a tabela de tokens são armazenadas as seguintes informações:

linha, coluna, identificador, nome, tipo e código.

Tabela de Simbolos

Para a tabela de simbolos são armazenadas as seguintes informações:
código identificador, tipo, categoria e nome.

Compilação

gcc *.c -o <executavel> -lm -g

Execução do programa

./<executavel> <nome_arquivo>.jusm

Saída

Gera como saída dois arquivos, um com a tabela de token e outro com a tabela de simbolos. Além disso, no terminal apresenta os erros léxicos entrados, caso ocorrerem.