CES41 - Lab1: Analisador Léxico

Gabriel Martinz, Marina Moreira

1 Descrição da Implementação

A implementação do laboratório foi feita com base no exemplo de analisador léxico para a linguagem Tiny fornecido durante o curso.

Foi utilizado o Flex (arquivo `cminus.l`) para substituir a necessidade de implementar o autômato finito integralmente "à mão". A implementação da função de recebimento de *tokens* foi feita no próprio arquivo `cminus.l` por utilizar a função `yylex()` do Flex para receber os *tokens* definidos no próprio arquivo.

No arquivo `main.c` foi implementado o executável do compilador, recebendo o arquivo `.cminus` e retornando, em `stdout`, os *tokens* encontrados na análise léxica.

Importante destacar que as diretivas de pré-processamento que determinam se o compilador será de tipo *scanner-only* ou *parser-only* foram utilizadas. Tais diretivas foram extraídas do arquivo `main.c` de exemplo da implementação do compilador para linguagem Tiny e achamos oportuno manter o código relativo à análise sintática, já que essa será a próxima implementaçao que faremos no curso.

No arquivo *utils.c* tem-se a especificação do que mostrar na tela mediante o caracter que foi encontrado.

No arquivo *globals.h*, foram definidos os tokens que o scanner pode encontrar (e printar na tela segundo o arquivo *utils.h*) e considerar como válidos. Nesse aquivo, foi deixada a parte já referente à análise sintática que virá a seguir no curso.

2 Resultados Obtidos

2.1 Teste 1

```
    /* Um programa para calcular o mdc.
    Segundo o algoritmo de Euclides. */
int gcd (int u, int v){
    if (v == 0)
        return u;
    else
        return gcd(v, u-u/v*v);
```

```
    /* u-u/v*v == u mod v */
}
void main(void){
    int x; int y;
    x = input();
    y = input();
    output(gcd(x,y));
}
```

2.2 Resultado 1

C- COMPILATION: teste1.cminus
2:reserved word: int
2:IDENTIFIER, val = gcd
2:(
2:reserved word: int
2:IDENTIFIER, val = u
2:,
2:reserved word: int
2:IDENTIFIER, val = v
2:)
3:
4:reserved word: if
4:(
4:IDENTIFIER, val = v
4:==
4:NUMBER, val=0
4:)
5:reserved word: return
5:IDENTIFIER, val = u
5:;
6:reserved word: else
7:reserved word: return
7:IDENTIFIER, val = gcd
7:(
7:IDENTIFIER, val = v
7:,
7:IDENTIFIER, val = u
7:-
7:IDENTIFIER, val = u
7:/
7:IDENTIFIER, val = v
7:*
7:IDENTIFIER, val = v
7:)
7:;

```
9:
10:reserved word: void
10:IDENTIFIER, val = main
10:(
10:reserved word: void
10:)
11:
12:reserved word: int
12:IDENTIFIER, val = x
12:;
13:reserved word: int
13:IDENTIFIER, val = y
13:;
14:IDENTIFIER, val = x
14:=
14:IDENTIFIER, val = input
14:(
14:)
14:;
15:IDENTIFIER, val = y
15:=
15:IDENTIFIER, val = input
15:(
15:)
15:;
16:IDENTIFIER, val = output
16:(
16:IDENTIFIER, val = gcd
16:(
16:IDENTIFIER, val = x
16:,
16:IDENTIFIER, val = y
16:)
16:)
16:;
17:
17:EOF
```

2.3 Teste 2

```
/* Um programa para ordenação de
uma matriz com dez elementos. */
int x[10];
int minloc(int a[], int low, int high)
{
```

```
    int i;
    int x;
    int k;
    k = low;
    x = a[low];
    i = low + 1;
    while (i < high)
    {
        if (a[i] < x)
        {
            x = a[i];
            k = i;
        }
        i = i + 1;
    }
    return k;
}

void sort(int a[], int low, int high)
{
    int i;
    int k;
    i = low;
    while (i < high - 1)
    {
        int t;
        k = minloc(a, i, high);
        t = a[k];
        a[k] = a[i];
        a[i] = t;
        i = i + 1;
    }
}

void main(void)
{
    int i;
    i = 0;
    while (i < 10)
    {
        x[i] = input();
        i = i + 1;
    }
    sort(x, 0, 10);
    i = 0;
    while (i < 10)
```

```
    {
        output(x[i]);
        i = i + 1;
    }
}
```

2.4 Resultado 2

C- COMPILATION: teste2.cminus
2:reserved word: int
2:IDENTIFIER, val = x
2:[
2:NUMBER, val=10
2:]
2:;
3:reserved word: int
3:IDENTIFIER, val = minloc
3:(
3:reserved word: int
3:IDENTIFIER, val = a
3:[
3:]
3:,
3:reserved word: int
3:IDENTIFIER, val = low
3:,
3:reserved word: int
3:IDENTIFIER, val = high
3:)
4:
5:reserved word: int
5:IDENTIFIER, val = i
5:;
6:reserved word: int
6:IDENTIFIER, val = x
6:;
7:reserved word: int
7:IDENTIFIER, val = k
7:;
8:IDENTIFIER, val = k
8:=
8:IDENTIFIER, val = low
8:;
9:IDENTIFIER, val = x
9:=
9:IDENTIFIER, val = a

9:[
9:IDENTIFIER, val = low
9:]
9:;
10:IDENTIFIER, val = i
10:=
10:IDENTIFIER, val = low
10:+
10:NUMBER, val=1
10:;
11:reserved word: while
11:(
11:IDENTIFIER, val = i
11:¡
11:IDENTIFIER, val = high
11:)
12:
13:reserved word: if
13:(
13:IDENTIFIER, val = a
13:[
13:IDENTIFIER, val = i
13:]
13:¡
13:IDENTIFIER, val = x
13:)
14:
15:IDENTIFIER, val = x
15:=
15:IDENTIFIER, val = a
15:[
15:IDENTIFIER, val = i
15:]
15:;
16:IDENTIFIER, val = k
16:=
16:IDENTIFIER, val = i
16:;
17:
18:IDENTIFIER, val = i
18:=
18:IDENTIFIER, val = i
18:+
18:NUMBER, val=1
18:;
19:

20:reserved word: return
20:IDENTIFIER, val = k
20:;
21:
23:reserved word: void
23:IDENTIFIER, val = sort
23:(
23:reserved word: int
23:IDENTIFIER, val = a
23:[
23:]
23:,
23:reserved word: int
23:IDENTIFIER, val = low
23:,
23:reserved word: int
23:IDENTIFIER, val = high
23:)
24:
25:reserved word: int
25:IDENTIFIER, val = i
25:;
26:reserved word: int
26:IDENTIFIER, val = k
26:;
27:IDENTIFIER, val = i
27:=
27:IDENTIFIER, val = low
27:;
28:reserved word: while
28:(
28:IDENTIFIER, val = i
28:¡
28:IDENTIFIER, val = high
28:-
28:NUMBER, val=1
28:)
29:
30:reserved word: int
30:IDENTIFIER, val = t
30:;
31:IDENTIFIER, val = k
31:=
31:IDENTIFIER, val = minloc
31:(
31:IDENTIFIER, val = a

31:,
31:IDENTIFIER, val = i
31:,
31:IDENTIFIER, val = high
31:)
31:;
32:IDENTIFIER, val = t
32:=
32:IDENTIFIER, val = a
32:[
32:IDENTIFIER, val = k
32:]
32:;
33:IDENTIFIER, val = a
33:[
33:IDENTIFIER, val = k
33:]
33:=
33:IDENTIFIER, val = a
33:[
33:IDENTIFIER, val = i
33:]
33:;
34:IDENTIFIER, val = a
34:[
34:IDENTIFIER, val = i
34:]
34:=
34:IDENTIFIER, val = t
34:;
35:IDENTIFIER, val = i
35:=
35:IDENTIFIER, val = i
35:+
35:NUMBER, val=1
35:;
36:
37:
39:reserved word: void
39:IDENTIFIER, val = main
39:(
39:reserved word: void
39:)
40:
41:reserved word: int
41:IDENTIFIER, val = i

41:;
42:IDENTIFIER, val = i
42:=
42:NUMBER, val=0
42:;
43:reserved word: while
43:(
43:IDENTIFIER, val = i
43:¡
43:NUMBER, val=10
43:)
44:
45:IDENTIFIER, val = x
45:[
45:IDENTIFIER, val = i
45:]
45:=
45:IDENTIFIER, val = input
45:(
45:)
45:;
46:IDENTIFIER, val = i
46:=
46:IDENTIFIER, val = i
46:+
46:NUMBER, val=1
46:;
47:
48:IDENTIFIER, val = sort
48:(
48:IDENTIFIER, val = x
48:,
48:NUMBER, val=0
48:,
48:NUMBER, val=10
48:)
48:;
49:IDENTIFIER, val = i
49:=
49:NUMBER, val=0
49:;
50:reserved word: while
50:(
50:IDENTIFIER, val = i
50:¡
50:NUMBER, val=10

50:)
51:
52:IDENTIFIER, val = output
52:(
52:IDENTIFIER, val = x
52:[
52:IDENTIFIER, val = i
52:]
52:)
52:;
53:IDENTIFIER, val = i
53:=
53:IDENTIFIER, val = i
53:+
53:NUMBER, val=1
53:;
54:
55:
55:EOF

2.5 Teste 3

Nesse exemplo, pegamos um código de Bubble Sort simples em C, apenas para efeitos de scanner mesmo, e, para funcionar, retiramos as partes do código com caracteres que não seriam identificados.

```
    /* C program for implementation
    of Bubble sort*/

void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

/* A function to implement bubble sort*/
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        /* Last i elements are already
        in place*/
        for (j = 0; j < n - i - 1; j++)
```

```
            if (arr[j] > arr[j + 1])
                swap(arr[j], arr[j + 1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf(arr[i]);
}

/* Driver program to test above functions*/
int main()
{
    int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

2.6 Resultado 3

C- COMPILATION: teste3_bubblesort.cminus
3:reserved word: void
3:IDENTIFIER, val = swap
3:(
3:reserved word: int
3:*
3:IDENTIFIER, val = xp
3:,
3:reserved word: int
3:*
3:IDENTIFIER, val = yp
3:)
4:
5:reserved word: int
5:IDENTIFIER, val = temp
5:=
5:*
5:IDENTIFIER, val = xp
5:;
6:*
6:IDENTIFIER, val = xp
6:=

6:*
6:IDENTIFIER, val = yp
6:;
7:*
7:IDENTIFIER, val = yp
7:=
7:IDENTIFIER, val = temp
7:;
8:
11:reserved word: void
11:IDENTIFIER, val = bubbleSort
11:(
11:reserved word: int
11:IDENTIFIER, val = arr
11:[
11:]
11:,
11:reserved word: int
11:IDENTIFIER, val = n
11:)
12:
13:reserved word: int
13:IDENTIFIER, val = i
13:,
13:IDENTIFIER, val = j
13:;
14:IDENTIFIER, val = for
14:(
14:IDENTIFIER, val = i
14:=
14:NUMBER, val=0
14:;
14:IDENTIFIER, val = i
14:¡
14:IDENTIFIER, val = n
14:-
14:NUMBER, val=1
14:;
14:IDENTIFIER, val = i
14:+
14:+
14:)
17:IDENTIFIER, val = for
17:(
17:IDENTIFIER, val = j
17:=

17:NUMBER, val=0
17:;
17:IDENTIFIER, val = j
17:¡
17:IDENTIFIER, val = n
17:-
17:IDENTIFIER, val = i
17:-
17:NUMBER, val=1
17:;
17:IDENTIFIER, val = j
17:+
17:+
17:)
18:reserved word: if
18:(
18:IDENTIFIER, val = arr
18:[
18:IDENTIFIER, val = j
18:]
18:¿
18:IDENTIFIER, val = arr
18:[
18:IDENTIFIER, val = j
18:+
18:NUMBER, val=1
18:]
18:)
19:IDENTIFIER, val = swap
19:(
19:IDENTIFIER, val = arr
19:[
19:IDENTIFIER, val = j
19:]
19:,
19:IDENTIFIER, val = arr
19:[
19:IDENTIFIER, val = j
19:+
19:NUMBER, val=1
19:]
19:)
19:;
20:
23:reserved word: void
23:IDENTIFIER, val = printArray

23:(
23:reserved word: int
23:IDENTIFIER, val = arr
23:[
23:]
23:,
23:reserved word: int
23:IDENTIFIER, val = size
23:)
24:
25:reserved word: int
25:IDENTIFIER, val = i
25:;
26:IDENTIFIER, val = for
26:(
26:IDENTIFIER, val = i
26:=
26:NUMBER, val=0
26:;
26:IDENTIFIER, val = i
26:¡
26:IDENTIFIER, val = size
26:;
26:IDENTIFIER, val = i
26:+
26:+
26:)
27:IDENTIFIER, val = printf
27:(
27:IDENTIFIER, val = arr
27:[
27:IDENTIFIER, val = i
27:]
27:)
27:;
28:
31:reserved word: int
31:IDENTIFIER, val = main
31:(
31:)
32:
33:reserved word: int
33:IDENTIFIER, val = arr
33:[
33:]
33:=

33:
33:NUMBER, val=64
33:,
33:NUMBER, val=34
33:,
33:NUMBER, val=25
33:,
33:NUMBER, val=12
33:,
33:NUMBER, val=22
33:,
33:NUMBER, val=11
33:,
33:NUMBER, val=90
33:
33:;
34:reserved word: int
34:IDENTIFIER, val = n
34:=
34:IDENTIFIER, val = sizeof
34:(
34:IDENTIFIER, val = arr
34:)
34:/
34:IDENTIFIER, val = sizeof
34:(
34:IDENTIFIER, val = arr
34:[
34:NUMBER, val=0
34:]
34:)
34:;
35:IDENTIFIER, val = bubbleSort
35:(
35:IDENTIFIER, val = arr
35:,
35:IDENTIFIER, val = n
35:)
35:;
36:IDENTIFIER, val = printArray
36:(
36:IDENTIFIER, val = arr
36:,
36:IDENTIFIER, val = n
36:)
36:;

37:reserved word: return
37:NUMBER, val=0
37:;
38:
38:EOF