Relatório 03

Vinícius de Oliveira Peixoto Rodrigues (245294)

Agosto de 2022

Item (c)

O analisador léxico serve como um transpilador, convertendo código em uma linguagem arbitrária para código em C. A tabela abaixo lista as substituições mais simples feitas pelo lexer:

símbolo	**tradução em C**	**observação**
`{...}`	`/*...*/`	
`mod`	`%`	
`or`	`||`	
`and`	`&&`	
`:=`	`=`	
`<>`	`==`	
`><`	`!=`	
`><`	`!=`	
`><`	`!=`	
`var`		ignorado quando em uma linha isolada
`bar`		ignorado quando em uma linha isolada
`.`		ignorado

Além dessas substituições simples, há duas coisas importantes. A primeira é que o seguinte pattern:

tem a intenção de dar match em expressões da forma

```
program <program_name>(<params>);
```

apesar de consumir agressivamente qualquer linha que comece com

```
program[qualquer coisa](
```

Ao fazer match desse padrão, o programa ”abre”a definição de função `main()` `{`, que só é ”fechada”depois pela keyword `end`, que será substituída pela chave de fechamento (`}`). Desse modo, o programa inteiro se encontra dentro da `main()` transpilada.

Outro ponto importante é a declaração de variáveis inteiras, na seguinte forma:

```
var
    <varlist> : integer;
```

onde o lexer descarta (ignora) a keyword `var` e dá match segundo a regra

```
^.*integer;     ShuffleInt();
```

ou seja, consumindo a nova linha seguinte inteira (por isso o `^`) desde que ela termine em `integer`. A função `ShuffleInt()`, por sua vez, faz o parse da lista de argumentos (basicamente fazendo o prepend da keyword `int` e copiando caractere por caractere tudo que vem antes de `: integer;`) (por isso a condição de parada `yytext[i] != ':'`).

Item (d)

O programa cria a *start condition* `Palavra`:

```
%START Palavra
```

E cria também as labels `NovaLinha`, `Espaco` e `NovaPagina`. Em seguida, sempre que o lexer encontra uma dessas três ocorrências, incrementa o contador adequado. Sempre que for encontrado algo que não corresponde a nenhum dentre os três casos `NovaLinha`, `Espaco` e `NovaPagina`, o parser incrementa o contador de palavras e passa para o estado `Palavra`, onde ele consome caracteres até encontrar `NovaLinha`, `Espaco` ou `NovaPagina`. Quando isso ocorre, o parser retorna pro estado 0 (inicial) e o ciclo de consumir todos os `NovaLinha`/`Espaco`/`NovaPagina` subsequentes → trocar para o estado `Palavra` e consumir todos os caracteres até o próximo espaço/newline/newpage se repete até ser encontrado o EOF.

```
> ./p_d
teste 123 abc
teste1
a b c d e

Resultados:
1 pagina(s)
3 linha(s)
9 palavra(s)
31 caracter(es)
> echo "pagina1\fpagina2" | ./p_d

Resultados:
2 pagina(s)
1 linha(s)
2 palavra(s)
16 caracter(es)
```

Figura 1: Testes do lexer `p_d`

Item (e)

O programa dá match em números escritos em formato octal e os imprime em formato decimal, utilizando o `sscanf()` com o formatador `"%o"` para ler o número e guardar na variável `valor`, e o `printf("%d", valor)` para imprimi-lo em decimal.

```
> flex p_e.l
> gcc -o p_e lex.yy.c -lfl
> ./p_e
20
2
230
19
100
8
1000
64
```

Figura 2: Testes do lexer `p_e`

Itens (f) e (g)

Os scripts se encontram em anexo.

```
> echo "hoje eh 31/08/2022" | ./p_g
hoje eh trinta e um de agosto de dois mil e vinte e dois
> echo "hoje eh 12/04/21" | ./p_g
hoje eh doze de abril de dois mil e vinte e um
> ./p_g < input_p_g
Hoje é dezenove de agosto de dois mil e quinze , dia de inicio da atividade 2 de EA872.
Amanhã será vinte de agosto de dois mil e quinze.
A próxima aula será em vinte e seis de agosto de dois mil e quinze.
~/faculdade/ea872/02-lex/lex main !4 ?8                                    14:07:28
>
```

Figura 3: Testes do lexer `p_g`