Aula 3 - Análise Léxica

Analisador léxico

Recebe como entrada o programa fonte e gera como saída uma sequência de tokens.

Uma das formas de construir um analisador léxico é escrever um diagrama que ilustre a estrutura dos tokens da linguagem fonte e o traduzir manualmente com um programa que os identifica.

- Pode realizar também tarefas secundárias a nível de interface com o usuário, como a remoção de espaços e comentários por exemplo.

Interação entre o analisador léxico e o parser

Tokens, padrões e lexemas

- Tokens são símbolos terminais da linguagem fonte (por exemplo if, else, dígitos, caracteres, operadores, etc.);
- Lexema são conjuntos de caracteres que são reconhecidos como um token.
- Um padrão descreve o conjunto de lexemas que podem representar um token em particular (quando há mais de um lexema possível para gerar um token).

Atributos para tokens

- Quando um token pode estar associado a dois ou mais lexemas, o analisador léxico deve prover atributos para os tokens para as fases subsequentes, para que ela possa distingui-los.

Exemplos:

- Em tokens numéricos, o valor do número representado pelo lexema pode ser o atributo do token.
- Para identificadores, o próximo lexema pode ser o atributo do token (`int numero`).

Erros léxicos

Determinados erros não podem ser detectados em nível léxico, como por exemplo o erro abaixo.

```
fi (a==f(x)) {
    ...
}
```

Os erros léxicos mais comuns são aqueles onde o analisador léxico não consegue associar o prefixo lido a nenhum dos padrões associados aos tokens da linguagem.

Nesse ponto, o analisador pode abortar a leitura, emitindo uma mensagem de erro, ou tratar o erro de alguma maneira.

Há quatro ações que configuram tentativas de recuperação de erros léxicos:

- Remover um caractere estranho da entrada;
- Inserir um caractere ausente;
- Substituir um dos caracteres incorretos por um caractere correto;
- Transpor dois caracteres adjacentes.

  Se uma ou mais das ações acima conseguir tornar o prefixo em um token válido, o analisador pode indicar ao usuário a sequência de ações como sugestão de correção do programa fonte, ou mesmo prosseguir assumindo esta correção.

Buferização

- O acesso à entrada pode ser o gargalo, em termos de performance, do compilador.
- A buferização consiste no uso de um ou mais vetores auxiliares (buffers) que permitem a leitura da entrada em blocos, de modo que o analisador léxico leia os caracteres a partir destes buffers, os quais são atualizados e preenchidos à medida do necessário, o que reduz o acesso ao disco.

Exemplo:

A) Consideraremos que a entrada cabe no buffer. Entrada: s = “ABCDE” Tamanho do buffer: N = 8

A	B	C	D	E	$
↑					
pos					

```
int get()
{
    if (buffer[pos]==EOF) throw new Exception("Erro");
    auto c = buffer[pos++];
    return c;
}
void unget()
{
    if (!pos) throw new Exception("Posição igual a zero.");
    --pos;

}
```

B) Considerando o *buffer* único quando a entrada é maior que o tamanho do buffer: s = “ABCDEFGHIJKLM” N = 8

0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	
								↑
								pos

```
int get()
{
    if (pos==N)
    {
        fread(buffer, N, sizeof(char), stdin);
        pos = 0;
    }
    auto c = buffer[pos++];
    return c;
}

int unget()
{
    if (!pos) pos=N-1;
    else --pos;

}
```

C) *Buffer* duplo s = "ABCDEFGHIJ" |s| = 10 N = 4

0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G		
								↑
								pos

```
int get(){
    if(pos==2*N and last_update==1){
        fread(buffer, N, sizeof(char), stdin); //atualiza a primeira metade
        pos=0;
        update=0;
    } else if (pos==N and last_update==0){
        fread(buffer+N, N, sizeof(char), stdin); //atualiza a segunda metade
        update=1;
    }
    if (pos==2*N) pos=0;
    return buffer[pos++];
}
void unget()
```

```
{
    if((pos==2*N or pos==N) and update==1)
        throw Exception(...);
    if(pos==N and update==0)
        throw Exception(...);
    pos = (pos==0 pr pos==2*N) ? 2*N-1 : pos-1;
}
```

Estratégias para implementação de analisadores léxicos

Pares de buffers

Ponteiros L e R

- Os tokens podem ser extraídos do par de *buffers* por meio do uso de dois ponteiros L e R.
- Uma cadeia de caracteres delimitada por estes dois ponteiros é o lexema atual.
- Inicialmente, os dois ponteiros apontam para o primeiro caractere do próximo lexema a ser identificado, então o ponteiro R avança até que o padrão de um token seja reconhecido.
- Daí, o lexema é processado e ambos ponteiros se movem para o primeiro caractere após o lexema.

Atualização dos *buffers* e o ponteiro R

Sentinelas

Definições

Um alfabeto, ou classe de caracteres, é um conjunto finito de símbolos.

- Exemplos: ASCII, EBCDIC, alfabeto binário {0,1}, os dígitos decimais, etc.
- Uma cadeia sobre um alfabeto
  Alpha é uma sequência finita de elementos de
  Alpha. Os termos sentença, palavra e string são geralmente usados como sinônimos de cadeia.
- Observe que as cadeias “AB” e “BA” são diferentes.

Conceitos associados à cadeias

- O comprimento de uma cadeia s é denotado por |s|;
- $|\epsilon| = 0$;
- Um prefixo de s é uma cadeia obtida pela remoção de zero ou mais caracteres do fim de s;

- Um sufixo de s é uma cadeia obtida pela remoção de zero ou mais caracteres do início de s;
- Uma subcadeia de s é uma cadeia obtida pela remoção de um prefixo e de um sufixo de s;
- Um prefixo, sufixo ou subcadeia de s é dito próprio se difere de ϵ e de s;
- Uma subsequência de s é uma cadeia obtida pela remoção de zero ou mais símbolos de s, não necessariamente contíguos.

  Uma linguagem é um conjunto de cadeias sobre algum alfabeto *Alpha* fixo.
- Observe que linguagens como $\emptyset$ e $\{\epsilon\}$ são contempladas por essa definição.

Operações em cadeias

- Concatenação (justaposição);

Operações em linguagens

Expressões regulares

Expressões regulares e parênteses

Propriedades das expressões regulares

Definições regulares Exemplo: identificadores em C/C++

$ id \leftarrow (letra | _)|| (letra|digito|_)*$

Classe de caracteres de um número na forma hexadecimal:

1. hex
   $leftarrow[0-9A-Fa-f]^+$: constantes hexadecimais não pré-fixadas com 0x.
2. hex
   $leftarrow(0(x|X))?[0-9A-Fa-f]^+$: constantes hexadecimais pré-fixadas ou não com 0x.