UFRGS – Informática - INF01147 - Compiladores - 2024/2
Prof. Marcelo Johann

Trabalho Prático - Especificação da Etapa 1:
Análise Léxica e Inicialização de Tabela de Símbolos

Resumo:

O trabalho consiste na implementação de um compilador funcional. Esta primeira etapa do trabalho consiste em fazer um analisador léxico utilizando a ferramenta de geração de reconhecedores *flex* e inicializar uma tabela global de símbolos encontrados em linguagem C++ com o *container* ***map*** de *STL* (*Standard Template Library*).

Funcionalidades necessárias:

A sua análise léxica deve fazer as seguintes tarefas:

a. reconhecer as expressões regulares que descrevem cada tipo de lexema;
b. classificar os lexemas reconhecidos em *tokens* retornando as constantes definidas no arquivo `tokens.h` fornecido ou códigos *ascii* para caracteres simples;
c. incluir os identificadores e os literais (inteiros, reais, caracteres e *strings*) em uma tabela de símbolos global implementada como um *container* com acesso eficiente por chave alfanumérica;
d. controlar o número de linha do arquivo fonte, e fornecer uma função declarada como `int getLineNumber(void)` a ser usada nos testes e pela futura análise sintática;
e. ignorar comentários de única linha e múltiplas linhas;
f. informar erro léxico ao encontrar caracteres inválidos na entrada, retornando o *token* de erro;
g. definir e atualizar uma variável global e uma função `int isRunning(void)`, que mantém e retorna valor *true* (diferente de 0) durante a análise e muda para *false* (igual a 0) ao encontrar a marca de fim de arquivo;

Descrição dos tokens

Existem tokens que correspondem a caracteres particulares, como vírgula, ponto-e-vírgula, parênteses, para os quais é mais conveniente usar seu próprio código *ascii*, convertido para inteiro, como valor de retorno que os identifica. Para os *tokens* compostos, como palavras reservadas e identificadores, cria-se uma constante (`#define` em C ANSI) com um código maior do que 255 para representá-los.

Os *tokens* representam algumas categorias diferentes, como palavras reservadas, operadores de mais de um caractere e literais, e as constantes definidas no código do trabalho são

precedidas por um prefixo para melhor identificar sua função, separando-as de outras constantes que serão usadas no compilador.

Palavras reservadas

As palavras reservadas da linguagem são: `char`, `int`, `if`, `then`, `else`, `while`, `read`, `print`, `return`. Para cada uma deve ser retornado o *token* correspondente.

Caracteres especiais

Os caracteres simples empregados pela linguagem são listados abaixo (estão separados apenas por espaços), e devem ser retornados com o próprio código *ascii* convertido para inteiro. Você pode fazer isso em uma única regra léxica que retorna `yytext[0]`. São eles:

```
, ; : ( ) [ ] { } = + - * / % < > & | ~
```

Identificadores

Os identificadores são usados para designar variáveis, vetores e nomes de funções, são formados por uma sequência de um ou mais caracteres alfabéticos minúsculos ou maiúsculos ou também dígitos, e não podem conter o caractere *underscore* (´_´);

Literais

Literais são formas de descrever constantes no código fonte. Literais inteiros são formados por uma sequência de um ou mais dígitos decimais após o símbolo '#'. Literais do tipo caractere são representados por um único caractere entre *aspas simples* (mais precisamente apóstrofo, ASCII decimal 39), como por exemplo: `'a'`, `'X'`, `'-'`. Literais do tipo *string* são quaisquer sequências de caracteres entre aspas duplas, como por exemplo `"meu nome"` ou `"E-mail!"`, e servem apenas para imprimir mensagens com o comando `"print"`. *Strings* consecutivas não podem ser consideradas como apenas uma, o que significa que o caractere de aspas duplas não pode fazer parte de uma *string*. Para incluir os caracteres de aspas duplas e final de linha dentro da *string*, devem ser usadas sequências de escape, como "\"" e "\n".

Comentários

Comentários de uma única linha começam em qualquer ponto com a sequência “//“ e terminam na próxima marca de final de linha, representada pelo caractere ‘\n’. Comentários de múltiplas linhas iniciam pela sequência “/**” e terminam pela sequência “**/”, sendo que podem conter quaisquer caracteres, que serão todos ignorados, incluindo uma ou mais quebras de linha, as quais, entretanto, devem ser contabilizadas para controle do número de linha.

Caracteres em branco

Os caracteres de espaço, tabulação e nova linha são considerados como “caracteres em branco” e serão ignorados pelo analisador léxico da linguagem. Portanto, eles podem ocorrer entre quaisquer outros lexemas, e serão usados apenas para definir a disposição visual no editor e para separar diferentes lexemas entre si.

Controle e organização do seu código fonte

Você deve manter o arquivo `tokens.h` intacto, e separar a sua função main em um arquivo especial chamado `main.cpp`, já que a função main não pode estar contida no código de `scanner.l`. Isso é necessário para permitir a automação dos testes, que utilizará uma função `main` especial escrita pelo professor, substituindo a que você escreveu para teste e desenvolvimento. Você deve usar essa estrutura de organização, manter os nomes `tokens.h` e `scanner.l`. Instruções mais detalhadas sobre o formato de submissão do trabalho e cuidados com detalhes específicos estão em outro documento separado.

Atualizações e Dicas

Verifique regularmente os documentos e mensagens da disciplina para informar-se de alguma eventual atualização que se faça necessária ou dicas sobre estratégias que o ajudem a resolver problemas particulares. Em caso de dúvida, consulte o professor.

Porto Alegre, 30 de Setembro de 2024