Implementação de um analisador sintático

Atividade II

Universidade Federal do ABC

Descrição

Neste segundo projeto você deve desenvolver uma gramática e o analisador sintático para a mesma. Você deverá ter como ponto de partida os *tokens* utilizados na atividade I, porém, pode implementar uma divisão maior dos *tokens* se necessário. Isso significa que você pode, por exemplo, ao ler um *token* de PALAVRA_RESERVADA recategorizá-los em T_IF (no caso de if) e T_VOID (no caso do tipo void). Lembre-se que o lexema também era identificado no arquivo de tokens. Seu código receberá uma lista de *tokens* no mesmo formato que o analisador léxico da atividade I gerava (ou seja, valores separados por @) e deve verificar se o código pode ser gerado pela gramática da sua linguagem. Note que nesse arquivo só estarão sendo utilizados os *tokens* definidos antes, a especialização mencionada deve ser feita por você na leitura dos *tokens* no analisador sintático.

Para facilitar, segue a lista de *tokens* definida na atividade I:

Token	ER
INTEIRO	[0-9]+
PONTO_FLUTUANTE	([0-9]*.[0-9]+([eE][+-]?[0-9]+)?|[0-9]+.[0-9]*)
STRING	“[^"\n]*”
BOOL	(true|false)
IDENTIFICADOR	[a-zA-Z][a-zA-Z0-9_]*
PALAVRA_RESERVADA	(while|if|else|return|int|bool|double|void)
PONTO_VIRGULA	;
ABRE_PARENTESES	(
FECHA_PARENTESES	)
ABRE_CHAVES	{
FECHA_CHAVES	}
VIRGULA	,
ATRIBUICAO	=
OPERADOR	(&&|||+|-|*|/|%)
OPERADOR_RELACIONAL	(==|<=|>=|!=|<|>)
INCREMENTO	(++|--)

Seu analisador sintático deve utilizar a abordagem **descendente preditiva recursiva**, isso significa que sua gramática deve ser LL(1) para que ele possa reconhecê-la. A descrição de como funciona um analisador sintático deste tipo foi apresentada nos slides *aula05_asd_p1.pdf*. Sugiro que você consulte os exemplos para te ajudar na implementação. **Sua gramática deve ter geração de condicionais, expressões aritméticas, funções e fazer uso de todos os tokens**. Note que, a gramática LL(1) para expressões aritméticas já está descrita nos slides (número 39) e pode te auxiliar no desenvolvimento do restante. Seu programa receberá (como argumento pela linha de comando) o caminho de um arquivo com os *tokens* e deverá imprimir na saída padrão “OK” se o código pode ser gerado pela sua gramática e mensagens de erro indicando o contrário. Quanto mais descritiva a mensagem de erro, melhor. **Se você implementar a geração da árvore de análise sintática corretamente você terá 2 pontos adicionais no projeto**. Nesse caso, a árvore de análise sintática deve ser impressa no término da execução de forma legível e a estrutura de dados utilizada deve ser descrita no relatório. Reforço que a geração da árvore é **opcional**.

Sua implementação deve ser feita em linguagem C. A equipe de desenvolvimento deverá ter no máximo 4

integrantes. Assuma que cada linha do arquivo de *tokens* terá no máximo 1024 caracteres e que cada lexema não passará de 200 caracteres. Em conjunto com o código, deverá ser entregue um relatório de 4 páginas (no máximo) descrevendo a gramática e as principais funções implementadas. O trabalho deverá ser entregue até o dia **29/07** . O uso de geradores de analisadores sintáticos (Yacc, Bison etc) está **proibido**. A entrega deve ser feita via **TIDIA**, sendo a submissão composta por **três arquivos**: um arquivo PDF com o relatório, um arquivo C com a implementação do analisador sintático e um arquivo fonte da sua linguagem que faça uso de **TODAS** as regras da gramática.

Seu código deve compilar sem apresentar nenhum **erro** ou **aviso** com a seguinte linha de comando:

```
gcc -o sintatico ARQUIVO.c -std=c99 -Wall -g
```