Diago Piccoli
Valmor Secco

Gramática

S <- main {**A**}
A <- **BA** | if(ident **H** opRel **GF**) {**A**} **A** | while(ident **H** opRel **GF**) {**A**} **A** | for(**B** ident **H** opRel **GF**; ident = **G**) {**A**} **A** | &
B <- tipo ident **C** | ident **C**
C <- = **D** | ;
D <- ident **E** | valor **E**
E <- opArit **D** | ;
F <- opLog ident **H** opRel **GF** | &
G <- ident **H** | valor **H**
H <- opArit **G** | &

	FIRST	FOLLOW
S	main	$
A	tipo, ident, if, while, for, &	$, }
B	tipo, ident	tipo, ident, if, while, for, }, $
C	“=, ;”	$
D	ident, valor	$
E	opArit, ;	$
F	opRel, &	$, ;)
G	ident, valor	$, opLog,), ;
H	opArit	$, opRel

Tabela Sintática

	main	if	while	for	tipo	}	ident	;	“="	valor	opArit	opLog	)	opRel	$
S	S <- main {A}														
A		A <- if (ident H opRel GF) {A} A	A <- while (ident H opRel GF) {A} A	A <- for (B ident H opRel GF ; ident = G) {A} A	A <- BA	A <- &	A <- BA								A <- &
B					B <- tipo ident C		B <- ident C								
C								C <- ;	C <- D						
D							D <- ident E			D <- valor E					
E								E <- ;			E <- opArit D				
F								F <- &				F <- opLog ident H opRel GF	F <- &		F <- &
G							G <- ident H			G <- valor H					
H											H <- opAirt G			H <- &	H <- &

Tecnologias

O projeto de compiladores foi desenvolvido utilizando PHP na versão >= 7 FPM que utiliza fast CGI para execução mais rápida de scripts. Utilizando o docker para criação de containers com o docker-compose para facilitar na criação e comunicação entre os containers. Utilizando NGINX como web server para ser possível a execução do PHP.
O projeto em si que está localizado na pasta site e tem como página inicial uma interface para a edição do código-fonte a ser compilado com um log do Lexer e um log do Parser. Além que executar um alerta em caso de encontrar algum erro ao compilar. Após editar o código e salvar ele faz uma atualização em um arquivo de texto temporário e executa o processo de Lexer, Parser e Semantic.
Dentro da pasta site/src estão localizados os arquivos de compilação divididos em Lexer, Parser, Semantic, SymbolTable, Token e Util. A pasta do Lexer armazena os códigos que fazem a identificação dos tokens e que armazenam a corrente de tokens. A pasta do parser trabalha com a corrente de tokens gerados sobre a gramática aplicada. A pasta do semantic faz uma análise das variáveis, verifica o tipo correto e se a mesma foi previamente declarada. A pasta de SymbolTable trabalha com a instância e identificação dos tokens a partir da expressão regular determinada por cada token. A pasta de token contém todas as classes de tokens trabalhados com o id, nome e expressão regular de cada um.
Após ler o arquivo temporário é passado ao Lexer a tabela de símbolos e o texto do arquivo, assim é criada a corrente de tokens e a mesma é enviada para o parser e para o semantic. Para ser validada a gramática com a análise sintática e posteriormente a análise semântica. Por fim é retordado um array com o texto original, o resultado do lexer e os logs do parser ou uma exceção caso exista algum erro.
Para executar o compilador é necessário ter o docker e o docker compose instalado na máquina e executar via console o comando:
docker-compose -f docker-compose-dev.yml up

*** O comando docker-compose executa o arquivo para criação dos containers, o -f indica o arquivo .yml a ser executado e up significa que está subindo a aplicação.**