Bison

Bison: Introdução

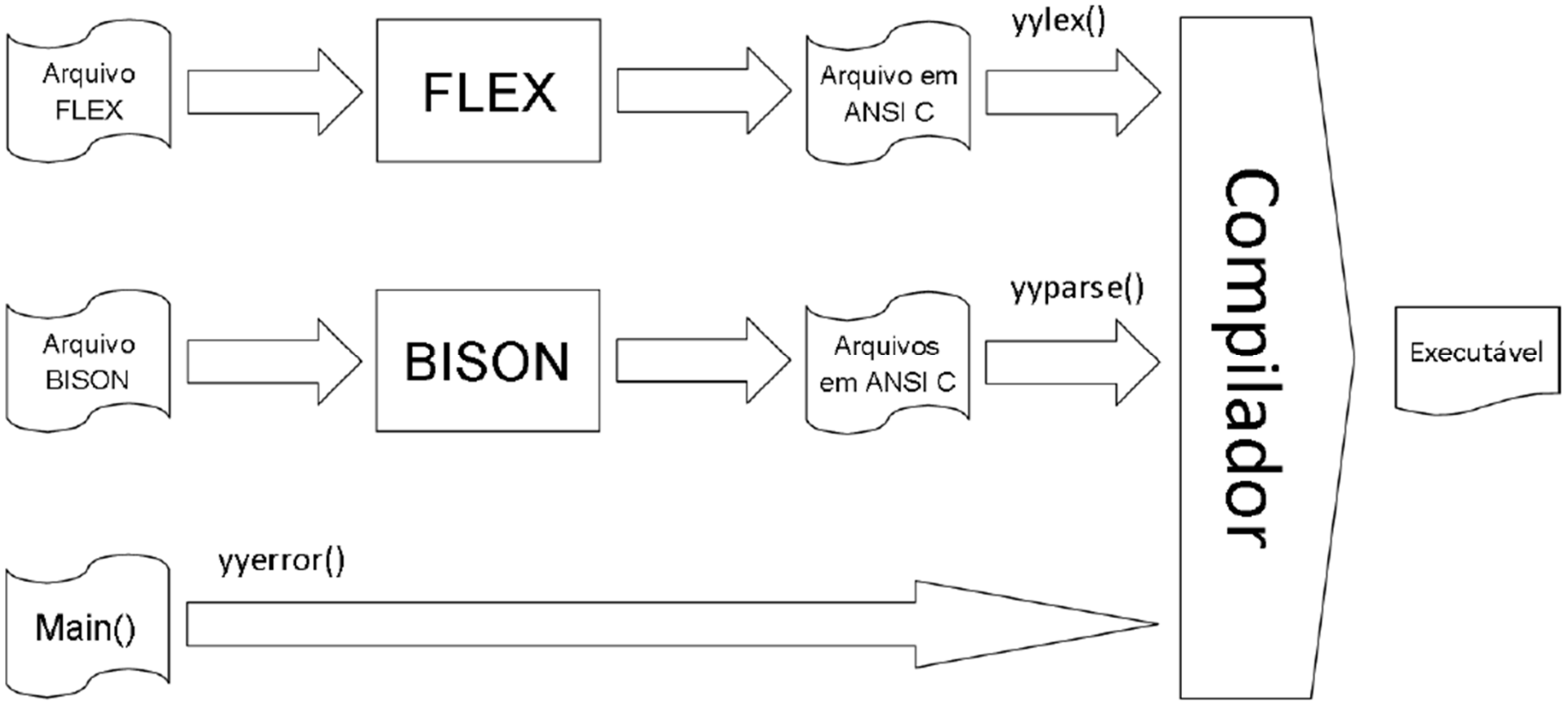
- O **Bison** é uma ferramenta que tem o objetivo de gerar o código de um programa para reconhecer a estrutura gramatical de uma entrada.

- O bison é a evolução do programa yacc**.**

- Foi originalmente desenvolvido para a construção de compiladores, sendo utilizando na geração de **analisadores sintáticos**.

- O Bison recebe como entrada, basicamente, uma sequência de produções de uma gramática livre de contexto; o que fazer quando um produção é reconhecida (ações); e produz como saída uma *parser* LALR.

Bison: Geração do Sintático em Conjunto com o Flex



Bison: Estrutura do Arquivo de Entrada

Os arquivos de entrada possuem, em geral, a extensão **`.y`** e são constituídos de 3 seções, delimitadas pelos caracteres **%%**.

Definições [opcional]

%%

Regras {ação} [padrão]

%%

Código Auxiliar em C/C++ [opcional]

Bison: Estrutura do Arquivo de Entrada

- A seção "***Definições***" é utilizada para a definição de macros e porções de código em C/C++. Toda porção de código C/C++ deve estar delimitado por ***%{*** e ***%}***.

- As "***Regras***" são destinadas a fornecer as regras gramaticais de todos os símbolos não-terminais e terminais, descrição da procedência de operadores além de tipos de dados dos valores semânticos dos diversos símbolos da gramática. As regras gramaticais definem a construção de cada símbolo não-terminal a partir dos *tokens* que o compõem.

- A seção "***Código Auxiliar em C/C++***" pode ser utilizada opcionalmente para descrever rotinas auxiliares, porém, no caso do programa escrito para o Bison ser independente, é necessário que essa seção contenha a função ***main()***.

Bison: Regras Gramaticais

Estrutura:

```
simbolo:definição { ação }
       |definição { ação }
       |          { ação }
;
```

- Dois pontos (:) separam o lado esquerdo do lado direito da regra, e o ponto-e-vírgula (;) finaliza a regra.
- A barra vertical “|” mostra as possibilidades de derivação para um mesmo não-terminal.
- Cada regra pode ter uma ação em associada a ela. Essa ação está entre chaves ({}).

Bison: Exemplo -> lexico.l

```
%option noyywrap
%{
#include <sintatico.tab.h>
int yylval;
%}

%%

"+" { return ADD; }
"-" { return SUB; }
"*" { return MUL; }
"/" { return DIV; }
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
\n     { return EOL; }
[ \t]  { /* ignore whitespace */ }
. { printf("Caracter Misterioso: %c\n", *yytext); exit(0);}

%%
```

Bison: Exemplo -> sintatico.y

```
%{
#include <stdio.h>
extern int yylex();
extern char* yytext;
%}
%token NUMBER
%token ADD
%token SUB
%token MUL
%token DIV
%token EOL

%start calclist
%%
calclist: exp EOL { printf("= %d\n", $1); return;}
;
exp: factor  { $$ = $1;}
             | exp ADD factor { $$ = $1 + $3; }
             | exp SUB factor { $$ = $1 - $3; }
;
factor: term { $$ = $1;}
             | factor MUL term { $$ = $1 * $3; }
             | factor DIV term { $$ = $1 / $3; }
;
term: NUMBER   { $$ = $1;}
;

%%
int main(int argc, char** argv)
{
    yyparse();
    return 0;
}
```

Bison: Geração do Sintático

```
$ flex arquivo.l
$ bison -d arquivo.y
$ gcc *.c -I. -o programa
```

O programa gerado já pode ser executado:

```
$ ./programa < arquivo_de_entrada
```

A opção **-d** no bison é para a geração de um arquivo de cabeçalho a ser utilizado pelo flex.