CSE443 Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Today

- Class will focus on PR02:
  - structure of Bison's .y file
  - yylex and yyparse
  - the union
  - symbol tables (read esp. section 2.7.1)
  - general advice

Structure of Bison's .y file

http://dinosaur.compilertools.net/bison/bison_6.html#SEC34

"A Bison grammar file has four main sections, shown here with the appropriate delimiters:

```
%{
C declarations
%}

Bison declarations

%%
Grammar rules
%%

Additional C code
```

Comments enclosed in '/* … */' may appear in any of the sections."

grammar.y

```
%{
#include <stdio.h>

/* EXTERN DECLARATIONS */
extern char * yytext;
extern int yylex();

/* FORWARD DECLARATIONS */
void yyerror(const char* p);

%}

/* DIRECTIVES */
%error-verbose


/* TOKENS */
%token ID 101

/* ASSOCIATIVITY AND PRECEDENCE DECLARATIONS */
%right ...
%left  low precedence operators
%left ...
%left  high-precedence operators


/* SYNTAX TREE NODE TYPE DECLARATIONS */
%union{
   struct Basic basic;
   struct ConstantValue k;
   struct ExpressionTypeInfo t;
}

%type <basic> ID
%type <k> C_INTEGER
%type <t> expression

%start program

%%

 /* GRAMMAR RULES W/ACTIONS */

program
   : definition_list sblock  {}
   ;

%%

void yyerror(const char* p){
    // do something reasonable
}
```

```
%{
C declarations
%}

Bison declarations

%%
Grammar rules
%%

Additional C code
```

yylex and yyparse

yylex is defined in lexer by Flex, called by yyparse.

yyparse is defined in parser by Bison, called by your code.

"the union"

```
/* SYNTAX TREE NODE TYPE DECLARATIONS */
%union{
   struct Basic basic;
   struct ConstantValue k;
   struct ExpressionTypeInfo t;
}

%type <basic> ID
%type <k> C_INTEGER
%type <t> expression
```

other possible unions

```
enum ConstantType { POINTER, INTEGER, BOOLEAN, CHARACTER, STRING };

struct ConstantValue {
   struct SymbolTableEntry * actualType;
   int lineNo;
   int colNo;
   enum ConstantType type;
   union {
      void * ptr;
      int i;
      bool b;
      char c;
      char * s;
   } value;
   ...
   ...
};
```

```
void printConstantValue(FILE * destination, struct ConstantValue * constant) {
  if (constant != NULL) {
    switch (constant -> type) {
    case POINTER:
      fprintf(destination,":= %p", constant->value.ptr);
      break;
    case INTEGER:
      fprintf(destination,":= %d", constant->value.i);
      break;
    ...
   default:
      internal_compiler_error("illegal variant used in ConstantValue");
    }
  }
}
```

Suggestive - your code need not do exactly this.