Bison Introduction

Compiler Design WS 17/18

Lab Assignment 2

- Implement RTSL Parser

Lab Assignment 2

- Implement RTSL Parser
- Output: List of syntactic elements, like CLASS, FUNCTION_DEF, DECLARATION, etc.
- Detect syntactic errors and some semantic errors

Lab Assignment 2

- Implement RTSL Parser
- Output: List of syntactic elements, like CLASS, FUNCTION_DEF, DECLARATION, etc.
- Detect syntactic errors and some semantic errors

```
testN.rtsl  :  Input file
testN.out   :  Expected stdout
testN.err   :  Expected stderr
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;
```

CLASS Test1 of type material

```
float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;
```

CLASS Test1 of type material

```
float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
EXPRESSION_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
FUNCTION_DEF shade
```

```
class Test1 : rt_Material;

float foo(int i) {
  if(i<0)
    return 0.0;
  else
    return 1.0;
}


void shade() {
  int i;
  float f;

  i = 0;
  f = foo(i);

  rt_SampleColor = color(f);
}
```

```
CLASS Test1 of type material
RETURN_STATEMENT
RETURN_STATEMENT
IF_ELSE_STATEMENT
FUNCTION_DEF foo
DECLARATION i of type int
DECLARATION f of type float
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
EXPRESSION_STATEMENT
FUNCTION_DEF shade
```

```
%token IF
%token ELSE

%%

if_statement:
   IF '(' expression ')' statement
 | IF '(' expression ')' statement ELSE statement
;

expression: /* ... */ ;
statement: /* ... */ ;

%%

/* ... */
```

```
%token IF
%token ELSE

%%

if_statement:
   IF '(' expression ')' statement
     { printf("IF_STATEMENT\n"); }
 | IF '(' expression ')' statement ELSE statement
     { printf("IF_ELSE_STATEMENT"); }
;

expression: /* ... */ ;
statement: /* ... */ ;

%%

/* ... */
```

```
%token IDENTIFIER TYPE

%%

var_declaration:
   type_specifier IDENTIFIER ';'
;

type_specifier:
   TYPE
 | /* ... */
;

%%

/* ... */
```

```
%token IDENTIFIER TYPE

%%

var_declaration:
   type_specifier IDENTIFIER ';'
     { printf("DECLARATION %s of type %s\n", "???", "???"); }
;

type_specifier:
   TYPE
 | /* ... */
;

%%

/* ... */
```

```
%token<str> IDENTIFIER TYPE

%%

var_declaration:
   type_specifier IDENTIFIER ';'
     { printf("DECLARATION %s of type %s\n", $2, "???"); }
;

type_specifier:
   TYPE
 | /* ... */
;

%%

/* ... */
```

yylval.str = ... in the lexer

```
%token<str> IDENTIFIER TYPE

%%

var_declaration:
   type_specifier IDENTIFIER ';'
     { printf("DECLARATION %s of type %s\n", $2, "???"); }
;

type_specifier:
   TYPE
 | /* ... */
;

%%

/* ... */
```

```
%token<str> IDENTIFIER TYPE
%type<str> type_specifier

%%

var_declaration:
    type_specifier IDENTIFIER ';'
      { printf("DECLARATION %s of type %s\n", $2, $1); }
;

type_specifier:
    TYPE { $$ = $1; }
  | /* ... */
;

%%

/* ... */
```