Exercise Sheet

09/30

1. 请设计以下 C 语言片段的词法与语法分析规则，并用 Flex 与 Bison 实现。编程中可以使用 AI 编程，使用 AI 生成测试用例，但是应当理解 AI 所写的代码与测试用例。需要提交源代码和测试用例。评分根据源代码和测试用例评分。使用 Bison 语法分析器不应当产生冲突。代码根据能否正确编译以及能否通过所有测试用例评分。测试用例根据能否分辨不正确的程序评分。

   【A】基本要求

   - 标识符（用作变量名、函数名、类型名等）：字母或下划线开头、仅包含字母、数字和下划线；
   - 空白：空格、制表符、换行都是应当跳过的空白；
   - 注释：没有注释；
   - 整数常量：支持十进制、以 `0` 开头的八进制、以 `0x/0X` 开头的十六进制。可不带后缀，或带 `u/U` 、`l/L` 、`ll/LL` ；无符号后缀可与一种长度后缀组合，顺序不限，如 `0x2ALL` 、`1UL` 、`1lu` ；
   - 浮点数：十进制形式，可带 `f/F/l/L` 后缀，如 `1.5` 、`.5` 、`1.` 等；
   - 字符与字符串：字符常量用单引号，字符串常量用双引号可以为空字符串；包含表示引号、反斜杠、换行和制表符的三种转义字符。

   【B】类型

   以下是全部基本类型

   - void 类型 `void`
   - 字符 `char`
   - 无符号字符 `unsigned char`
   - 短整数 `short` 、`short int`
   - 无符号短整数 `unsigned short` 、`unsigned short int`
   - 整数 `int`
   - 无符号整数 `unsigned int`
   - 长整数 `long` 、`long int`
   - 无符号长整数 `unsigned long` 、`unsigned long int`
   - 长长整数 `long long` 、`long long int`
   - 无符号长长整数 `unsigned long long` 、`unsigned long long int`
   - 单精度浮点 `float`
   - 双精度浮点 `double`
   - 扩展精度浮点 `long double`

在此基础上，还应支持以下符合类型：数组、指针、函数指针、枚举、结构体、联合体。

【C】强制类型转换与 sizeof 参数中的类型

以下是一些语法正确的例子

```
( int * )p;
sizeof(int);
sizeof(int [3]);
sizeof(int ( * )[3]);
sizeof(int ( * )(int x));
```

【D】函数头与函数声明

以下例子都是合法的函数头或者函数声明

- `int add(int x, int y)` 普通的二元函数，既可以用作函数头，又可以用作函数声明（需要加分号，后同）；
- `int add(int, int)` 只能用作函数声明；
- `int * get(int i)` 返回值是指针的函数，既可以用作函数头，又可以用作函数声明；
- `int f()` 和 `int g(void)` 都是合法的无参数函数声明和函数头；
- `int ( * fp )(int x)` 既不能做函数头又不能做函数声明，这是一个函数指针的声明；
- `int apply(int ( * callback )(int x), int y)` 以函数指针为第一个参数，既可以用作函数头，又可以用作函数声明；
- `int apply(int ( * callback )(int), int y)` 同样既可以用作函数头，又可以用作函数声明；
- `int apply(int ( * )(int), int)` 只能用作函数声明。
- 以下这些都是涉及数组的函数，既可以用作函数头，又可以用作函数声明

```
int sum(int a[10])
int sum(int a[], int n)
int sum_rows(int ( * a )[3], int n)
```

【E】初始化

以下都是合法的例子

```
int a = 1, b = 2 + 3;
int values1[] = {1, 2, 3,};
( char * ) values2[] = {( char * ) 0, ( char * ) values2};
int matrix1[2][2] = {{1, 2}, {3, 4}};
int matrix2[2][2] = {1, 2, 3, 4};
struct Point { int x, y; } point = {1, 2};
```

语法分析时，只需得到花括号构成的树结构即可，不需要处理这个树结构与类型之间的对应关系。

【F】结构体与联合体

以下都是合法的例子

```
struct Node;
struct Node {
  int value, count;
  int data[3];
  struct Node *next;
  struct { int x, y; } point;
};
struct Node node;
struct { int x, y; } point = {1, 2};
```

换言之，允许结构体声明和结构体定义，结构体的定义可以是匿名的，结构体定义可以在结构体或联合体的域定义时一边定义一遍使用。定义、声明或使用结构体时必须使用关键字 `struct` 。联合体 `union` 的定义也是类似的。

【G】枚举

以下都是合法的例子

```
enum Color { RED, GREEN = 5, BLUE };
enum Color color = GREEN;
```

【H】表达式

- 下标 `a[i]` 、成员 `a.x` `p->x` 、后缀 ‘++ –‘
- 前缀 `++` `--` 、强制类型转换
- 运算（优先级不同）：`*` 、`/` 、`%` 、`+` 、`-`
- 位移：`<<` 、`>>`
- 大小比较：`<` 、`<=` 、`>` 、`>=`
- 相等比较：`==` 、`!=`
- 按位运算（优先级不同）：按位与 `&` 、按位异或 `^` 、按位或 `|`
- 逻辑运算（优先级不同）：逻辑与 `&&` 、逻辑或 `||`
- 赋值运算（右结合）：`=` 、`*=` 、`/=` 、`%=` 、`+=` 、`-=` 、`<<=` 、`>>=` 、`&=` 、`^=` 、`|=`
- 逗号连接多个表达式

注意：`*p->ptr` 表达式中的优先级。

【I】语句

- 基本语句: 空语句、表达式语句、含声明和语句的块
- 条件：`if` , `else` 、`switch` 、`case` （只能接常数和标识符）、`default`
- 循环: `while` 、`do while` 、`for` （不允许初始化时边声明边使用）
- 跳转：`break` 、`continue` 、`return`

【J】程序

程序包含变量声明（可能含初始化）、结构体声明与定义、联合体声明预定义、函数声明与定义。