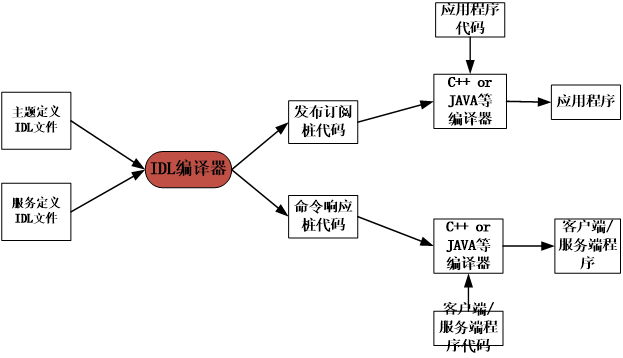
## 实验一：使用Antlr4开源工具构建MIDL语言编译器前端

### MIDL简介

IDL（Interface Definition Language）是一种平台无关的接口定义（或描述）语言。IDL主要用于描述分布式通信的数据接口，它的语法结构简单、功能强大、跨平台，是分布式数据通信应用开发的最佳选择。IDL编译技术完成两项工作：

1. **生成发布订阅需要的数据结构和桩文件；**
2. **生成命令响应过程中的桩代码**。

IDL语言使得嵌入式中间件通信所需的数据接口和具体实现语言分离，通过IDL编译器代码生成工具，生成相应语言的接口代码，可大大提高应用程序开发效率。



MIDL（Mini Interface Definition Language）是IDL语言的一个语法子集，作为编译原理课程的实验对象。

### MIDL的词法语法规则描述

**其词法规则描述如下：**

1. MIDL语言的关键字（20个）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| module | struct | boolean | short | long |
| unsigned | int8 | int16 | int32 | int64 |
| uint8 | uint16 | uint32 | uint64 | char |
| string | float | double | true | false |

注：关键字是保留字，并且不区分大小写。

1. 语言的专用符号20个：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| { | } | [ | ] | ; |
| , | + | - | / | \* |
| % | >> | << | = | & |
| | | ~ | ^ | : | :: |

1. 标示符ID、整数INTEGER、浮点数FLOATING\_PT、单字符CHAR、字符串STRING和BOOLEAN等词法规则（标黑的）通过下列正则表达式定义：

LETTER = [a-z] | [A- Z]

DIGIT = [0-9]

UNDERLINE= \_

ID = LETTER (UNDERLINE?( LETTER | DIGIT))\*

INTEGER\_TYPE\_SUFFIX = l | L

INTEGER = (0 | [1-9] [0-9]\*) INTEGER\_TYPE\_SUFFIX?

EXPONENT=( e | E) ( + | - )? [0-9]+

FLOAT\_TYPE\_SUFFIX= f | F | d | D

FLOATING\_PT= [0-9]+ . [0-9]\* EXPONENT? FLOAT\_TYPE\_SUFFIX?

| . [0-9]+ EXPONENT? FLOAT\_TYPE\_SUFFIX?

| [0-9]+ EXPONENT FLOAT\_TYPE\_SUFFIX?

| [0-9]+ EXPONENT? FLOAT\_TYPE\_SUFFIX

ESCAPE\_SEQUENCE = \ ( b | t | n | f | r | " | ' | \ )

CHAR= '(ESCAPE\_SEQUENCE | (~\ | ~') )'

STRING = " (ESCAPE\_SEQUENCE | (~\ | ~") )\* "

BOOLEAN = TRUE | true | FALSE | false

补充说明的注意事项

其中，正则符号标识含义如下表：

|  |  |
| --- | --- |
| ~\ | ~" | 表示匹配除了 \ ，"外所有字符 |
| [a-z] | 表示a字符到z字符，与课件ppt中含义一致 |
| ? | 表示匹配 0 或 1 次，与课件ppt中含义一致 |
| + | 表示匹配 1 或 多次，与课件ppt中含义一致 |
| \* | 表示匹配 0 或 多次，与课件ppt中含义一致 |
| a | b | 表示匹配 a字符或b字符，与课件ppt中含义一致 |
| ( ) | 表示一个组，比如()\*表示\*对括号内组整体起作用，与课件ppt中含义一致 |

空白、换行符和制表符在词法分析时可以忽略掉，但是字符串本身的空白、换行符和制表符不能忽略。

**MIDL的语法规则描述如下：**

specification -> definition { definition}

definiton -> type\_decl“**;**”| module “**;**”

module -> “**module**”ID “**{**” definition { definition } “**}**”

type\_decl -> struct\_type | “**struct**” ID

struct\_type->**“struct”** **ID “{”** member\_list **“}”**

member\_list-> { type\_spec declarators **“;”** }

type\_spec -> scoped\_name | base\_type\_spec | struct\_type

scoped\_name -> [**“::”**] ID {**“::”** ID}

base\_type\_spec->floating\_pt\_type|integer\_type|**“char”**|**“string”**|

**“boolean”**

floating\_pt\_type -> **“float”** | **“double”** | **“long double”**

integer\_type -> signed\_int | unsigned\_int

signed\_int->(**“short”**|**“int16”**)

|(**“long”**|**“int32”**)

|(**“long”** **“long”**|**“int64”**)

|**“int8”**

unsigned\_int -> (**“unsigned”“short”**| **“uint16”**)

| (**“unsigned”“long”**| **“uint32”**)

| (**“unsigned” “long” “long”** | **“uint64”**)

| **“uint8”**

declarators -> declarator {**“,”** declarator }

declarator -> simple\_declarator | array\_declarator

simple\_declarator -> ID [**“=”** or\_expr]

array\_declarator -> ID “[” or\_expr “]” [**“=”** exp\_list ]

exp\_list -> **“[”** or\_expr { **“,”**or\_expr } **“]”**

or\_expr -> xor\_expr {**“|”** xor\_expr }

xor\_expr -> and\_expr {**“^”** and\_expr }

and\_expr -> shift\_expr {**“&”**shift\_expr }

shift\_expr -> add\_expr { (**“>>”** | **“<<”**) add\_expr }

add\_expr -> mult\_expr { (**“+”** | **“-”**) mult\_expr }

mult\_expr -> unary\_expr { (**“\*”** |**“/”**|**“%”**) unary\_expr }

unary\_expr -> [**“-”**| **“+”** | **“~”**] literal

literal -> **INTEGER** | **FLOATING\_PT** | **CHAR** | **STRING** | **BOOLEAN**

**注意事项：**

1. 红色双引号表示匹配字符串本身
2. 红色{ }表示括起来的内容可以重复0至n次
3. 红色[ ]表示括起来的内容可选，即存在不存在都合法
4. 空白、换行符和制表符在词法分析时忽略掉。

### Antlr4开源工具简介

Antlr4（Another Tool for Language Recognition）是一款基于Java开发的开源的语法分析器生成工具，能够根据语法规则文件生成对应的语法分析器，广泛应用于DSL构建，语言词法语法解析等领域。Antlr可以生成不同target的AST，包括Java、C++、JS、Python、C#等，可以满足不同语言的开发需求。下面是Antlr4相关网址：

官方网址： <https://www.antlr.org/>

Github开源地址：<https://github.com/antlr/antlr4>

官方文档：<https://github.com/antlr/antlr4/blob/master/doc/index.md>

### 实验一任务要求

1. 根据Antlr4开源工具简介内容，自行搜索学习Antlr4的使用方法，主要涉及1）如何再G4文件中定义词法语法的规则。2）如何使用Antlr4工具解析G4生成相应的词法，语法分析程序。3）如何调用词法，语法分析程序中的方法生成对应MIDL源代码的抽象语法树。
2. 完成MIDL词法语法规则的G4文件定义。
3. 使用Antlr4编译G4文件生成MIDL语言词法，语法分析程序的源代码。
4. 根据MIDL语法规则的文法给出相应的抽象语法树结构，提交相应的语法树定义的文件,语法树的定义尽量简洁易懂，必要时提供说明。
5. 使用生成的词法，语法分析程序构建一个从MIDL源代码到抽象语法树的分析程序，并输出格式化的抽象语法树到SyntaxOut.txt文件中。
6. 给出测试方法描述,提交readme.doc,如果你还有其它需要说明的问题须写在readme.doc中。
7. 编程语言不限制，目前Antlr4支持的target有Java, C#, Python2|3, JavaScript, Go, C++, Swift, Dart, PHP，只要在这些语言范围之内都可以。