附录 A

C 语言语法

C语言的ISO标准的附件B给出了一套完整的语言语法规则。本附录再现了这些规则，而且我把这些规则编写得更易读[①]。在每条规则中，语法项的名称会出现在左侧并以黑体字的形式显示出来。符号 | 、*、+、[、]、(和)具有下列的含义：

- 项目$_1$ | 项目$_2$表示项目$_1$和项目$_2$可以两者选其一。
- 项目*表示项目可以重复零次或多次。
- 项目+表示项目可以重复一次或多次。
- [项目]表示项目是可选的。
- (和)用于可选择项组。

但是，当把这些符号设置成为**courier粗体**时，它们具有通常C语言中的含义。虽然大多数规则相当清楚，但是有一些需要更深入的解释说明。如果需要，还会有注释。

A.1 记号

记号 关键字 | 标识符 | 常量 | 字符串字面量 | 运算符 | 标点符号

预处理 头文件名 | 标识符 | 预处理数 | 字符常量 | 字符串字面量 | 运算符 | 标点符号 | 每种不属于上述字符的非空白字符

> “记号”是组成程序不可分割的符号。预处理器识别一些编译器不识别的记号，因此记号和预处理记号之间有区别。

A.2 关键字

关键字 **auto** | **break** | **case** | **char** | **const** | **continue** | **default** | **do** | **double** | **else** | **enum** | **extern** | **float** | **for** | **goto** | **if** | **int** | **long** | **register** | **return** | **short** | **signed** | **sizeof** | **static** | **struct** | **switch** | **typedef** | **union** | **unsigned** | **void** | **volatile** | **while**

A.3 标识符

标识符 非数字 (非数字 | 数字)*

非数字 **_** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z** | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z**

数字 **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

A.4 常量

常量 浮点常量 | 整型常量 | 枚举常量 | 字符常量

浮点常量 小数常量 [指数部分] [浮点后缀] | 数字+ 指数部分 [浮点后缀]

小数的常量 数字* **.** 数字+ | 数字+ **.**

指数部分 (**e** | **E**) [**+** | **-**] 数字+

① 这些资料经ANSI许可改编自American National Standards Institute ANSI/ISO 9899©1990。这个标准的副本可从ANSI购买（ANSI，11 West 42nd Street, New York, NY 10036）。

浮点后缀　**f** | **l** | **F** | **L**

默认情况下，浮点常量是以double格式存储的。在浮点常量末尾的字母f或F通知编译器要以float型存储常量。l或L则通知编译器以long double型存储常量。

整型常量　十进制常量 [整数后缀] | 八进制常量 [整数后缀] |
十六进制常量 [整数后缀]

十进制常量　非零数字 数字*

八进制常量　**0** 八进制数字*

注意，0是正式区分作为八进制常量的，不是十进制常量。当然，这个特殊的举措没有什么差异，因为0在任何情况下的含义相同。

十六进制常量　(**0x** | **0X**) 十六进制数字+

非零的数字　**1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

八进制数字　**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7**

十六进制数字　**1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **a** | **b** | **c** | **d** | **e** | **f** | **A** | **B** | **C** | **D** | **E** | **F**

整数后缀　无符号后缀 [长后缀] | 长后缀 [无符号后缀]

588

无符号后缀　**u** | **U**

长后缀　**l** | **L**

整型常量末尾的字母u或U通知编译器把常量作为unsinged int型来存储。l或L则通知编译器把常量作为long int型来存储。当常量后跟有两种字母时（顺序任意），都把常量作为unsigned long int型来存储。

枚举常量　标识符

枚举常量用于枚举元规则中（见A.11节）。

字符常量　**'***c*字符+**'** | **L'***c*字符+**'**

如果出现了L，常量表示的是宽字符。

c字符　除去**'**、****以外的任何字符，或换行符 | 转义序列

小心正确地解释上述的规则。规则说明字符常量不包含换行符，但是它没说明字符常量不能包含'或\字符。这两者始终会作为转义序列的内容出现在字符常量中。

转义序列　简单转义序列 | 八进制转义序列 | 十六进制转义序列

简单转义序列　**\'** | **\"** | **\?** | **** | **\a** | **\b** | **\f** | **\n** | **\r** | **\t** | **\v**

八进制转义序列　**** 八进制数字 [八进制数字] [八进制数字]

十六进制转义序列　**\x** 十六进制数字+

由于历史原因，八进制转义序列限定为三个数字。另一方面，十六进制转义序列可以是任意数量的数字。

A.5　字符串字面量

字符串字面量　**"***s*字符***"** | **L"***s*字符***"**

如果出现了L，字符串字面量就为宽字符串。

s字符　除去**"**、****以外的任何字符，或换行符 | 转义序列

这条规这不表示字符串常量不能含有"或\字符。这两者始终会作为转义序列的内容出现在字符常量中。

A.6　运算符

运算符　**[** | **]** | **(** | **)** | **.** | **->** | **++** | **--** | **&** | ***** | **+** | **-** | **~** | **!** | **sizeof** | **/** | **%** | **<<** | **>>** | **<** | **>** | **<=** | **>=** | **==** | **!=** | **^** | **|** | **&&** | **||** | **?** | **:** | **=** | ***=** | **/=** | **%=** | **+=** | **-=** | **<<=** | **>>=** | **&=** | **^=** | **|=** | **,** | **#** | **##**

为了方便，把预处理器运算符#和##也包含到C语言的普通运算符组中了。

A.7　标点符号

标点符号　**[** | **]** | **(** | **)** | **{** | **}** | ***** | **,** | **:** | **=** | **;** | **...** | **#**

一些标点符号也是运算符，这要依赖于实际内容。例如，记号=当用在声明中时是标点符号，主要是为了把变量和它的初始符分离开，或者是为了把枚举常量和它的值分离开。而当记号=用于表达式时，它就是赋值运算符。记号...（省略号）用于写带有可变长度实参列表的函数。 589

A.8　头文件名

头文件名　**<** *h*字符$^+$ **>** | **"***q*字符$^+$**"**

*h*字符　除了换行符和**>**的任何字符。

*q*字符　除了换行符和**"**的任何字符。

头文件名几乎可以包含几无任何字符。允许如此灵活的原因是头文件名常常含有依赖操作系统的信息（例如路径）。

A.9　预处理数

预处理数　[**.**] 数字 (数字 | 非数字 | (**e** | **E**) (**+** | **-**) | **.**)*

在预处理期间，对要监测的数应用这条简单规则，这条规则允许一些不合法的数（比如0x0y）溜掉。但是，这些不合法的数稍后会由编译器检查出来，所以不会有害的。

A.10　表达式

基本表达式　标识符 | 常量 | 字符串字面量 | **(** 表达式 **)**

基本表达式是不可分割的表达式。不是因为它是单独的标识符、常量或字符串字面量，就是因为它是用括号闭合的。所有其他的表达式都服从于C语言的优先级和结合性规则，这些规则都嵌入到随后的19条规则中了。

后缀表达式　基本表达式 (**[** 表达式 **]** | **(** [参数表达式列表] **)** | **.** 标识符 | **->** 标识符 | **++** | **--**)*

参数表达式列表　赋值表达式 (**,** 赋值表达式)*

为了避免作为参数分隔符的逗号标点和逗号运算符的混淆，函数调用中的实际参数必须是“赋值表达式”，而不能是任意表达式。

一元表达式　(**++** | **--** **sizeof**)* (后缀表达式 | 一元运算符 强制类型转换表达式 | **sizeof (**类型名**)**)

一元运算符　**&** | ***** | **+** | **-** | **~** | **!**

强制类型转换表达式　(**(** 类型名 **)**)* 一元表达式

乘法类表达式　强制类型转换表达式 ((***** | **/** | **%**)强制类型转换表达式)*

加法类表达式　乘法类表达式 ((**+** | **-**) 乘法类表达式)*

移位表达式　加法类表达式 ((**<<** | **>>**) 加法类表达式)*

关系表达式　移位表达式 ((**<** | **>** | **<=** | **>=**) 移位表达式)*

判等表达式　关系表达式 ((**==** | **!** **=**) 关系表达式)*

与表达式　判等表达式 (**&** 判等表达式)*

异或表达式　与表达式 (**^** 与表达式)* 590

或表达式　异或表达式 (**|** 异或表达式)*

逻辑与表达式　或表达式 (**&&** 或表达式)*

逻辑或表达式　逻辑与表达式 (**||**逻辑与表达式)*

条件表达式　逻辑或表达式 (**?** 表达式 **:** 条件表达式)*

赋值表达式　(一元表达式 赋值运算符)* 条件表达式

赋值运算符　= | *= | /= | %= | += | -= | <<= | >>= | &= | ^= | |=

表达式　赋值表达式 (, 赋值表达式)*

常量表达式　条件表达式

把常量表达式定义成条件表达式，而不是通常的表达式，这是因为C禁止在常量表达式中有赋值运算符和逗号运算符。（虽然语法规则中没有显示出来，但是C语言也不允许自增、自减运算符和函数调用。）

A.11　声明

声明　声明说明符 [初始声明符列表];

声明说明符　(存储类型说明符 | 类型说明符 | 类型限定符)+

前述的规则有些误导，因为它说明声明可以包含多于一个的存储类型说明符。实际上，只允许一个真正的存储类型，而且它必须在类型说明符和类型限定符之前。规则的正确理解是可以用typedef（由于语法目的所以考虑成是存储类型说明符）开始，后边跟着存储类型。类型说明符和类型限定符才是真的可以像规则显示的那样混合，这样会引发诸如int const unsigned volatile long这样的奇异组合。

初始声明符列表　初始声明符 (, 初始声明符)*

初始声明符　声明符 [= 初始化式]

存储类型说明符　typedef | extern | static | auto | register

为了简化语法规则，所以把typedef与真正的存储类型混在一起了。

类型说明符　void | char | short | int | long | float | double | signed | unsigned | 结构或联合说明符 | 枚举说明符 | 类型定义名

结构或联合说明符　(struct | union) (说明符 | [说明符] { 结构声明+ })

结构声明　说明符限定符列表 结构声明符列表 ;

说明符限定符列表　(类型说明符 | 类型限定符)+

结构声明符列表　结构声明符 (, 结构声明符)*

结构声明符　声明符 | [声明符] : 常量表达式

在前述的规则中常量表达式说明位域的宽度。如果出现了常量表达式，则可以忽略声明符，并产生一个未命名的位域。

枚举说明符　enum (标识符 | [标识符] { 枚举元列表 })

591

枚举元列表　枚举元 (, 枚举元)*

枚举元　枚举常量 [= 常量表达式]

类型限定符　const | volatile

声明符　(* 类型限定符*)* 直接声明符

直接声明符　(说明符 | (声明符)) ([[常量表达式]] | (参数类型列表) | ([标识符列表]))*

参数类型列表　参数声明 (, 参数声明)* [, ...]

在参数列表末尾出现的，...表明可能跟随额外的可变数量的参数。

参数声明　声明说明符 [声明符 | 抽象声明符]

标识符列表　标识符 (, 标识符)*

类型名　说明符限定符列表 [抽象符声明符]

类型名用于一元表达式和强制类型转换表达式规则中（见**表达式**）。

抽象声明符　(* 类型限定符*)+ | (* 类型限定符*)* 直接抽象声明符

普通声明符包含名字和关于名字性质的信息；而抽象声明符说明了性质，但是忽略名字。函数原型void f (int **, float[]);就使用了抽象声明符**和[]来帮助描述f的参数类型。

直接抽象声明符　(抽象声明符) | [(抽象声明符)] ([[常量表达式]] | ([参数类型列表]))+

类型定义名　标识符

初始化式　赋值表达式 | **{** *初始化式列表* [**,**] **}**
　　不，这不是错误。初始化式列表可以真的后边跟随（多余的）逗号。
初始化式列表　*初始化式* (**,** *初始化式*)*

A.12　语句

语句　标号语句 | 复合语句 | 表达式语句 | 选择语句 | 循环语句 | 跳转语句
标号语句　标识符 **:** 语句 | **case** 常量表达式 **:** 语句 | **default :** 语句
　　最后两种格式的标号语句只允许出现在switch语句中。
复合语句　**{** *声明** *语句** **}**
表达式语句　[表达式] **;**
　　由于语法目的，空语句被看成是表达式语句，这种表达式语句是缺少表达式的。
选择语句　**if (** 表达式 **)** 语句 [**else** 语句] | **switch (** 表达式 **)** 语句　592
　　虽然没有严格要求，但是switch语句体事实上始终是复合语句。虽然会忽略在声明中的初始化式，但是复合语句可以有声明。
循环语句　**while (** 表达式 **)** 语句 |
　　do 语句 **while (** 表达式 **) ;** |
　　for ([表达式] **;** [表达式] **;** [表达式] **)** 语句
跳转语句　**goto** 标识符 **;** | **continue ;** | **break ;** | **return** [表达式] **;**

A.13　外部定义

翻译单元　外部声明+
外部声明　函数定义 | 声明
函数定义　[声明说明符] 声明符 声明* 复合语句
　　声明说明符描述函数的返回类型。声明符给出了函数名和参数列表。声明（只出现在经典C风格的函数定义中）说明参数的类型。复合语句是函数体。

A.14　预处理指令

预处理文件　[组]
组　([预处理记号] 换行 | *if* 部分 | 控制行)+
*if*部分　*if* 组 *elif* 组* [*else*组] *endif* 行
*if*组　**# if** 常量表达式 换行 [组] |
　　# ifdef 标识符 换行 [组] |
　　# ifndef 标识符 换行 [组]
*elif*组　**# elif** 常量表达式 换行 [组]
*else*组　**# else** 换行 [组]
*endif*行　**# endif** 换行
控制行　**# include** 预处理记号 换行 |
　　# define 标识符 替换列表 换行 |
　　# define 标识符 **(** [标识符列表] **)** 替换列表 换行 |
　　# undef 标识符 换行 |
　　# line 预处理记号 换行 |
　　# error [预处理记号] 换行 |
　　# pragma [预处理记号] 换行 |
　　# 换行
lparen　没有前述空白的左圆括号字符。
替换列表　[预处理记号]
预处理记号　预处理记号+
换行　换行符　593