符号	对应终结符	符号	对应终结符	符号	对应终结符
++ -- ! ~ + -	ω_0	* / %	ω_1	+ -	ω_2
<< >>	ω_3	< <= > >=	ω_4	== !=	ω_5
= -= += ... ^=	ω_6	||	"||"	&&	'&&'
|	"|"	^	"^"	&	"&"
(	"("	)	")"	;	";"
常数	c	标识符	v	字符串	s
void,int,float,char	$type$	if	if	else	$else$
{	"{"	}	"}"	while	$while$
,	','				

初始文法

我们将实现的的文法设计如下：

Program -> **External_Declaration** | **Program External_Declaration**

External_Declaration -> **Function_Definition** [IDF()] {function entry}| **Declaration** ";"

Function_definition -> $type$ [addT] **Direct_declarator Compound_statement**

Direct_declarator -> v [IDF] '(' ')'

compound_statement -> '{' '}' | '{' **Statements**'}'

Statements -> **Statement** |**Statements Statement**

Statement -> **IfStatement** | **Assignment_expression** ";" [clr()]| **Declaration** ";" [clr()] | **While_Statement** | **compound_statement** | ';' [clr()]

Assignment_expression -> **logic_or_expression** ω_6 **Assignment_expression** {GEQ ω_6} | **logic_or_expression**

logic_or_expression -> **logic_or_expression** '||' **logic_and_expression** {GEQ || }| **logic_and_expression**

logic_and_expression -> **logic_and_expression** "&&" **or_expression** {GEQ &&}| **or_expression**

or_expression -> **or_expression** '|' **xor_expression** {GEQ |} | **xor_expression**

xor_expression -> **xor_expression** '^' **and_expression** {GEQ ^}| **and_expression**

and_expression -> **and_expression** '&' **equal_expression** {GEQ &}| **equal_expression**

equal_expression -> **equal_expression** ω_5 **relation_expression** {GEQ ω_5}|**relation_expression**

relation_expression -> **relation_expression** ω_4 **shift_expression** {GEQ ω_4}| **shift_expression**

shift_expression -> **shift_expression** ω_3 **add_expression** |{GEQ ω_3} **add_expression**

add_expression -> **add_expression** ω_2 **mul_expression** {GEQ ω_2}| **mul_expression**

mul_expression -> **mul_expression** ω_1 **unary_expression** {GEQ ω_1} | **unary_expression**

unary_expression -> ω_0 **unary_expression** {GEQ ω_0} | **T**

T -> c {push c} | v {push v} | s {push s}|'(' **Assignment_expression** ')'

IfStatement -> if "(" **E** ")" {IF(if)} **Statement** {IE(ie)}

| if "(" **E** ")" **Statement** $else$ {EL(el)} **Statement** {IE(ie)}

Declaration -> $type$ [add_type] **Variable_List**

Variable_List -> **Variable_List** ',' **init_declarator** | **init_declarator**

init_declarator -> v [IDV] | v "=" **Assignment_expression** [GEQ w6]

While_Statement -> $while$ {WH()} '(' **Expression** ')' {DO(do)} **Statement** {WE(we)}

其中，加粗字体表示为非终结符，花体字体与双引号内容表示为终结符。

LL1文法：

我们使用缩写将上述文法中的非终结符表示出来，其中{}内的内容表示select集，select集后的数字表示产生式的编号，其中使用[]括起来的内容属于动作函数。

P -> **ExD P1** $\{type\}$ **58** | ϵ {#} **67**

P1 -> **ExD P1** $\{type\}$ **59** | ϵ {#} **68**

ExD -> $tpye$ [add_type] v [add_name] **ExD1** $\{type\}$ **60**

ExD1 ->**Func1** { "(" } **61** | [IDV(v, type, len)] **Declaration1** ";" {"=", ";"} **62**

Func1 -> "(" ")" [IDF(v, type)] **CS** { "(" } **63**

Declaration1 -> w6 **AsE** [GEQ w6] **64** | ϵ {",", " ;"} **69**

CS -> '{' **CS1** '}' { '{' } **65**

CS1 -> **Ss** {ω_0, c , v , s, '(' , '{' , ';' , if, $while$, $type$ } **66** | ϵ { '}' } **70**

Ss -> **S Ss** {ω_0, c , v , s, '(' , '{' , ';' , if, $while$, $type$ } **0** | ϵ { '}' } **41**

S -> **AsE** ";" {ω_0, c , v , s, '(' } **2** | **IfS** $\{if\}$ **44** | **Declaration** ";"$\{type\}$ **45** | **Wh** $\{while\}$ **46** | ";" {;} **47** | **CS** { '{' }**48**

AsE -> **LoE** AsE_1 {ω_0, c , v , s, '(' } **4**

AsE_1 -> ω_6 **AsE** [GEQ ω_6] {ω_6} **5** |[deal] ω **LoE** {ω} **6** | ϵ {";",")" , ","} **42**

LoE ->**LaE** LoE_1 {ω_0, c , v , s, '(' } **7**

LoE_1 -> "||" **LaE** [GEQ ||] LoE_1 {"||"} **8**| "ϵ" {ω_6, ';' , ')', #}**9**

LaE -> **OrE** LaE_1 {ω_0, c , v , s, '(' }**10**

LaE_1 -> "&&" **OrE** LaE_1 [GEQ &&] {"&&"}**11** | ϵ {ω_6,"||" ,';' , ')', #}**12**

OrE -> **XoE** OrE_1 {ω_0, c , v , s, '(' }**13**

OrE_1-> "|" **XoE** [GEQ |] OrE_1 {"|"} **14** | ϵ {ω_6,"||", "&&" ,';' , ')', #}**15**

XoE -> **AnE** XoE_1 {ω_0, c , v , s, '(' }**16**

XoE_1 -> "^" **AnE** [GEQ ^] XoE_1 {"^"}**17** | ϵ {ω_6,"||", "&&", "|" ,';' , ')', #}**18**

AnE -> **EqE** AnE_1 {ω_0, c , v , s, '(' }**19**

AnE_1 -> "&" **EqE** [GEQ &] AnE_1 {"&"}**20** | ϵ {ω_6,"||", "&&", "|" ,"^" ,';' , ')', #}**21**

EqE -> **ReE** EqE_1 {ω_0, c , v , s, '(' } **22**

EqE_1 -> ω_5 **ReE** [GEQ ω_5] EqE_1 {ω_5}**23** | ϵ {ω_6,..."^", "&" ,';' , ')', #}**24**

ReE -> **ShE** ReE_1 {ω_0, c , v , s, '(' }**25**

ReE_1 -> ω_4 **ShE** [GEQ ω_4] ReE_1 {ω_4}**26** | ϵ {ω_6,..."&", ω_5 ,';' , ')', #}**27**

ShE -> **AdE** ShE_1 {ω_0, c , v , s, '(' }**28**

ShE_1 -> ω_3 **AdE** [GEQ ω_3] ShE_1 {ω_3}**29** | ϵ {ω_6,...ω_5 , ω_4 ,';' , ')', #}**30**

AdE -> **MuE** AdE_1 {ω_0, c , v , s, '(' }**31**

AdE_1 -> ω_2 **MuE**[GEQ ω_2] AdE_1 {ω_2}**32** | ϵ {ω_6,...ω_4 , ω_3 ,';' , ')', #}**33**

MuE -> **UnE** MuE_1 {ω_0, c , v , s, '(' } **34**

MuE_1 -> ω_1 **UnE** [GEQ ω_1] MuE_1 {ω_1} **35** | ϵ {ω_6,...ω_2 , ω_1 ,';' , ')', #}**36**

UnE -> ω_0 **UnE** [GEQ ω_0] {ω_0} **37** | **T** {c , v , s, '(' }**38**

T -> c [push c] | v [push v] |s [push s] **39** |'(' **AsE** ')' {"("}**40**

IfS -> if "(" **AsE** ")" [IF(if)] **S** [EL(el)] **IFS1** {if} [IE(ie)] **49**

IFS1 -> $else$ **S** {$else$} **50** | 'ϵ' **{**ω_0**,** c **,** v **,** s**, '(' , '{' , ';' ,** if**,** $while$**,** $type$ **}51**

Declaration -> $type$ [add_type] **VL** {$type$} **52**

VL -> v [IDV()] **VL1** {v} **53**

VL1 -> **Declaration1** **VL1** {"w6"} **54** | "," v [add_name] **VL1** {","} **55** | ϵ {";"} **56**

Wh -> $while$ [WH()] '(' **AsE** ')' [DO(do)] **S** [WE(we)] {$while$} **57**