错误处理说明

错误类型	错误类别码	解释	对应文法及出错符号
非法符号	a	格式字符串中出现非法字符报错行号为**<FormatString>**所在行数。	<FormatString> → “”{<Char>}””
名字重定义	b	函数名或者变量名在**当前作用域**下重复定义。注意，变量一定是同一级作用域下才会判定出错，不同级作用域下，内层会覆盖外层定义。报错行号为**<Ident>**所在行数。	<ConstDef>→<Ident> ... <VarDef>→<Ident> ... |<Ident> ... <FuncDef>→<FuncType><Ident> ... <FuncFParam> → <BType> <Ident> ...
未定义的名字	c	使用了未定义的标识符报错行号为**<Ident>**所在行数。	<LVal>→<Ident> ... <UnaryExp>→<Ident> ...
函数参数个数不匹配	d	函数调用语句中，参数个数与函数定义中的参数个数不匹配。报错行号为函数调用语句的**函数名**所在行数。	<UnaryExp>→<Ident>‘(’[FuncRParams]‘)’
函数参数类型不匹配	e	函数调用语句中，参数类型与函数定义中对应位置的参数类型不匹配。报错行号为函数调用语句的**函数名**所在行数。	<UnaryExp>→<Ident>‘(’[FuncRParams]‘)’
无返回值的函数存在不匹配的 return 语句	f	报错行号为**‘return’**所在行号。	<Stmt>→‘return’ {‘[’Exp’]’}‘;’
有返回值的函数缺少 return 语句	g	只需要考虑函数末尾是否存在 return 语句，**无需考虑数据流**。报错行号为函数**结尾**	FuncDef → FuncType Ident ‘(’ [FuncFParams] ‘)’ Block MainFuncDef → 'int' 'main' '(' ')' Block

		的'}'所在行号。	
不能改变常量的值	h	<LVal>为常量时，不能对其修改。报错行号为**<LVal>**所在行号。	<Stmt>→<LVal>'=' <Exp>';'|<LVal>'=' 'getint' '(' ')' ';'
缺少分号	i	报错行号为分号**前一个非终结符**所在行号。	<Stmt>,<ConstDecl>及<VarDecl>中的';'
缺少右小括号')'	j	报错行号为右小括号**前一个非终结符**所在行号。	函数调用(<UnaryExp>)、函数定义(<FuncDef>)及<Stmt>中的')'
缺少右中括号']'	k	报错行号为右中括号**前一个非终结符**所在行号。	数组定义(<ConstDef>,<VarDef>,<FuncFParam>)和使用(<LVal>)中的']'
printf 中格式字符与表达式个数不匹配	l	报错行号为**'printf'**所在行号。	Stmt →'printf''('FormatString{,Exp}')'';'
在非循环块中使用 break 和 continue 语句	m	报错行号为**'break'与'continue'**所在行号。	<Stmt>→'break'';'|'continue'';'

【注】错误 i，j，k 类型中的"前一个非终结符"强调的是在文法规则里出现在 ；）］之前的非终结符号，在分析中处理的是该非终结符产生的终结符号串的最后一个符号，也就是 ；）］本应该正常出现的位置的上一个单词

（3）**所有错误都不会出现恶意换行的情况**，包括字符、字符串中的换行符、函数调用等等。

（4）其他类型的错误，错误的行号以能够断定发现出错的第一个符号的行号为准。例如有返回值的函数缺少返回语句的错误，只有当识别到函数末尾的}时仍未出现返回语句，才可以断定出错，报错行号即为}的行号。