错误处理阶段设计文档

编码之前的设计思路

错误类型	错误类别码	解释及举例
非法符号或不符合词法	a	例如字符与字符串中出现非法的符号，符号串中无任何符号
名字重定义	b	同一个作用域内出现相同的名字（不区分大小写）
未定义的名字	c	引用未定义的名字
函数参数个数不匹配	d	函数调用时实参个数大于或小于形参个数
函数参数类型不匹配	e	函数调用时形参为整型，实参为字符型；或形参为字符型，实参为整型
条件判断中出现不合法的类型	f	条件判断的左右表达式只能为整型，其中任一表达式为字符型即报错，例如'a'==1
无返回值的函数存在不匹配的return语句	g	无返回值的函数中可以没有return语句，也可以有形如return;的语句，若出现了形如return(表达式);或return();的语句均报此错误
有返回值的函数缺少return语句或存在不匹配的return语句	h	例如有返回值的函数无任何返回语句；或有形如return;的语句；或有形如return();的语句；或return语句中表达式类型与返回值类型不一致
数组元素的下标只能是整型表达式	i	数组元素的下标不能是字符型
不能改变常量的值	j	这里的常量指的是声明为const的标识符。例如 const int a=1;在后续代码中如果出现了修改a值的代码，如给a赋值或用scanf获取a的值，则报错。
应为分号	k	应该出现分号的地方没有分号，例如int x=1缺少分号 （7种语句末尾，for语句中，常量定义末尾，变量定义末尾）
应为右小括号')'	l	应该出现右小括号的地方没有右小括号，例如fun(a,b;，缺少右小括号（有/无参数函数定义，主函数，带括号的表达式，if，while，for，switch，有/无参数函数调用，读、写、return）
应为右中括号']'	m	应该出现右中括号的地方没有右中括号，例如int arr[2;缺少右中括号（一维/二维数组变量定义有/无初始化，因子中的一维/二维数组元素，赋值语句中的数组元素）
数组初始化个数不匹配	n	任一维度的元素个数不匹配，或缺少某一维的元素即报错。例如int a[2][2]={{1,2,3},{1,2}}
<常量>类型不一致	o	变量定义及初始化和switch语句中的<常量>必须与声明的类型一致。int x='c';int y;switch(y){case('1') ....}
缺少缺省语句	p	switch语句中，缺少<缺省>语句。

以上是我们这次作业需要判断的错误类型。其中a是文法错误，其余的均为语法错误。

要完成本次作业，我觉得主要有两个步骤：第一个是建立符号表，第二个是根据不同类型错误可能出现的位置判断不同的错误类型。建立符号表和错误判断的过程我将在编码之后的设计思路中具体说明。

编码之后的设计思路

符号表的建立

我将符号表的建立与更新主要分为了两步：一种是添加整个层次，另一种是在当前层次下添加符号。

需要向符号表中增加层次主要有以下四种函数，我使用了函数名来作为该层次的key值。

- `<程序>`，暂时使用`Program`作为该层次的名称
- `<有返回值函数>`，使用函数名的标识符作为该层次的名称
- `<无返回值函数>`，使用函数名的标识符作为该层次的名称
- `<主函数>`，暂时使用`Main`作为该层次的名称

需要在同一层次向符号表中添加项的主要有如下情况。

- 常量定义（kind=0）：相对变量定义较为简单，只有整数和字符两种情况，不会有数组或者字符串。
  - `<程序>`中的`<常量说明>`
  - `<有返回值函数定义>`和`<无返回值函数定义>`中的`<复合语句>`中的`<常量说明>`
- 变量定义（kind=1）
  - `<程序>`中的`<变量说明>`
  - `<有返回值函数定义>`和`<无返回值函数定义>`中的`<复合语句>`中的`<变量说明>`
- 有返回值函数定义（kind=2）
  - `<程序>`中的`<有返回值函数定义>`

```
// 将该函数名添加到符号表
initCurrentSymbol();
curSym.kind = 2;
curSym.type = 0;
curSym.name = lastIdentifier;
insertCurrentSymbol();
```

- 无返回值函数定义（kind=3）
  - `<程序>`中的`<无返回值函数定义>`

```
// 将该函数名添加到符号表
initCurrentSymbol();
curSym.kind = 3;
curSym.type = 0;
curSym.name = lastIdentifier;
insertCurrentSymbol();
```

- 参数表（kind=4）

- `<有返回值函数定义>`和`<无返回值函数定义>`中的`<参数表>`

```
// 将参数添加到符号表中，仅有kind和type，没有value
initCurrentSymbol();
curSym.kind = 4;
if (lastTypeIdentifier == INTTK) {
    curSym.type = 1;
}
else {
    curSym.type = 4;
}
curSym.name = lastIdentifier;
insertCurrentSymbol();
```

- 主函数定义（kind=5）
  - `<主函数>`

```
// 将main添加到符号表
initCurrentSymbol();
curSym.kind = 5;
curSym.type = 0;
curSym.name = MAIN;
insertCurrentSymbol();
```

错误处理

错误类型a

错误类型a是本次作业中唯一的文法错误，所以需要在文法分析阶段进行处理。主要分为以下两个情况。

- 在有单引号包围的字符情况下：检查读取到的字符是否有非法字符。
- 在有双引号包围的字符串情况下，检查读取到的字符是否有非法字符；并且在读取到右双引号之后检查读取到的字符串是否为空集。

错误类型bc

要判断错误bc的前提是要建立符号表。

由于根据我们的文法，我们并不会有太多的函数嵌套，常量定义和变量定义也只有在`<程序>`、`<有返回值函数定义>`、`<无返回值函数定义>`中才会出现。因此在查询符号表的过程中，只需要查询当前所在函数的符号表和最外层的符号表（全局变量）即可。

所以我们在常量定义和变量定义的过程中向符号表中增加符号，在使用到标识符的地方查询符号表是否有相应的符号。

需要注意的是：建立符号表，如果是变量及初始化，每行仅能有一个；若变量无初始化，每行可以有多个。

错误类型de

在函数定义的`<参数表>`部分，我们可以更新该函数的参数表。在函数调用`<值参数表>`部分，我们可以将实际的输入参数和参数表中的需要的参数进行比对，进而识别并输出错误。

错误类型f

该错误可能发生的位置较为单一，在条件判断语句中判断等号两边的表达式是否为整型，进而识别并输出错误。

错误类型g和h

要判断函数的返回类型是否和定义相同，我们需要建立一个函数表。

当我们读取到`＜有返回值函数定义＞`的`＜声明头部＞`以及`＜无返回值函数定义＞`的`void＜标识符＞`部分，我们就可以判断出该函数的返回值类型和函数名称。因此我们在这里将函数名称和返回值类型加入到函数表中，当读取到`<语句列>`中的`<返回语句>`时与函数表进行比较判断，进而识别并输出错误。

错误类型i

该错误只会出现在<因子>和<赋值语句>中。在这些语句可能使用到数组的地方，如果接收到的数组下标不是整型，则识别并输出该型错误。

错误类型j

该错误只会出现在赋值语句和scanf语句中。在赋值语句和scanf语句出现标识符的地方，通过符号表查询是否为常量，从而识别错误并输出。

错误类型klm

该错误相对来说比较容易识别。在原来的语法分析的基础上，需要接收分号、右小括号和右中括号的地方，如果未能成功接收该符号，则捕捉并输出该错误。

需要注意的是，如果缺少的符号位于一行的最后一个，那么在输出错误的时候要注意行数需要定位到上一个单词所在的行数。以缺少分号为例，调用`getsym()`函数会获取到下一行的第一个单词，相应的行数计数器也会增加，但是我们缺少分号的错误行数应该是上一行。

错误类型o

该错误主要出现在变量定义及初始化和switch语句中的`<常量>`，需要判断等号两边的类型是否契合，否则捕捉o型错误。

错误类型p

该错误需要在`<情况语句>`中没能成功接受`DEFAULTTK`的symbol时捕捉。