错误分析阶段设计文档

输入输出要求

- 输入：`testfile.txt`
- 输出：`error.txt`
- 输出分析得到的错误行号以及类型

程序环境

- 语言：C++
- IDE：VS2019
- 测试环境：VS自带编译器、C++ 14、Windows10
- 目标环境：Clang 8.0.1、C++ 11、Linux

程序设计

符号表构造

- `IdentifyTable`类，

  设置为全局、静态类，整个程序通过直接调用其中设置为`static`的函数可以直接向符号表登记新的函数、常量、变量，或查询`IDENTIFY`标识符是否存在，或查询已登记的函数的参数表，常量、变量的类型。

  - `static void init()`

    初始化。

  - `static bool add_func(WordInfo* id, IdentifyType ret_type, ParameterTable* paras);`
    `static bool add_const(WordInfo* id, WordInfo* type);`
    `static bool add_var(WordInfo* id, WordInfo* type, int dimension);`

    向符号表添加新的符号信息。

  - `static bool have_var_const(WordInfo* id);`
    `static IdentifyType get_type_by_name(WordInfo* id);`
    `static IdentifyProperty get_property_by_name(WordInfo* id);`

    查询登记过的常量、变量信息。

  - `static bool have_func(WordInfo* func_id);`
    `static bool have_return(WordInfo* func_id);`
    `static bool check_func_para_num(WordInfo* func_id, ParameterValue* values);`
    `static bool check_func_para_type(WordInfo* func_id, ParameterValue* values);`
    `static IdentifyType get_return_type(WordInfo* func_id);`

    查询登记过的函数信息。

- `IdentifyBlock`类

  全局变量以及函数存储在全局`block`中，每增加一个作用域（推理到本次设置是每增加一个函数），则增加一个`block`存放该作用域中的符号名称。
  详细内容略。

- `IdentifyInfo`类

  存放每个符号的`WordInfo`信息，小写名称（因要求不区分大小写），属性（常量、变量或函数），类型（常量、变量为`int`或`char`，函数存返回值类型），函数参数表或常量、变量的维数。
  详细内容略。

错误表构造

- `Error`类
  登记每个错误信息，记录错误所在行号和类型。
  详细内容略。
- `ErrorTable`类

  设置为全局、静态类，整个程序通过直接调用其中设置为`static`的函数可以向表中增加新的错误、查询是否有错误被登记以及输出错误到文件。
  - `static void log_error(int line, string type)`

    登记新的错误。
  - `static void print_to_file(string file_name)`

    输出错误到文件。
  - `static bool have_error()`

    检查是否有错误被登记。

整体流程

1. 程序进行词法分析，若出现错误直接登记到错误表。
2. 程序进行语法分析，若出现错误直接登记到错误表。
3. 程序检查错误表是否为空，是则输出语法分析结果，否则输出错误内容，输出时首先对错误进行排序。

值得一提的错误及处理

错误1

- 错误：存在非`void`函数不是所有路径都有返回值。
- 表现：本地测试与测评机测试结果不一致。
- 分析：不设置返回值则返回值不确定，是危险的。亦可能不同编译器、编译环境对默认返回值的处理不同。
- 处理：增设返回值。

错误2

- 错误：对出现的错误信息按行号进行排序时，使用C++的`sort()`对`vector`中内容进行排序后排序结果不稳定。
- `sort()`的使用方法：在需要排序的`Error`类中对操作符"<"、">"、"=="、"!="、进行了定义。
- 表现：运行后排序结果有时升序有时降序。
- 处理：放弃使用`sort()`函数，手写冒泡排序。

错误3

- 错误：在测评机环境编译`translate()`函数出现问题。
- 表现：仅在测评机环境编译错误。
- 分析：`translate()`函数传入的`tolower`函数在Linux环境下有宏定义与该函数重名，导致编译器分析失败。
- 处理：更改命名空间设置让编译器成功链接到该函数。

需要分析的错误类型摘录

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。