编译原理实验二报告

程序功能

实现了对于C--语言的语义分析，并完成了选做要求2.1，2.2，2.3的内容。

程序解析第一个参数，将其作为输入文件打开并进行词法分析，语法分析，语义分析，如果输入文件有语义错误则打印对应的错误类型与错误提示。

由于使用到的数据结构较多，实验二的代码采用的是C++编写，以便于对于不同的对象的抽象，并利用extern "C"与实验一的代码进行对接，调用C语言的接口，获得实验一的语法树。

数据结构

定义的数据结构除了实验一中的语法树以外还有：Type，Field，Function，SymTable类。

首先是定义了Type为数据类型，其中的Kind有四个选择：Basic，Array，Struct，Error，代表基本类型，数组类型，结构体类型。其中Error类型是为了在探测到语义错误的时无法推断变量类型的特殊处理。Field为字段，其中包含了字段的类型和名字。Function为符号表中的函数项，其中包含了函数名字，返回类型，字段列表，以及为了支持函数声明与定义分离的辅助字段has_body与行号。

在遍历语法树的过程中在SymTable，内有三个字段，functionTable为函数表，fieldTableStack为字段表栈。structTable为结构体定义表。其中为了支持嵌套的字段定义，在FieldTable的基础上加入了栈结构以实现。几个表结构都是C++的hashmap，加速字符串结构的查询。

程序在完成语法分析后构造SymTable对象，然后调用SymTable的traverseAST函数，递归的对语法树进行分析。在SymTable类中对于每一个语法节点种类都有对应的方法，在分析过程中递归调用。

实现函数声明与定义分离

首先需要更改文法支持函数声明：

```
ExtDef:
      Specifier ExtDecList
      | Specifier SEMI
      | Specifier FunDec CompSt //函数定义
      | Specifier FunDec SEMI //函数声明
```

然后在处理ExtDef节点的时候遇到函数声明，那么就去查询对应的函数符号表，如果没有则添加一项，且设置has_body为false。如果有则调用eq方法判断两个声明是否相等，如果相等则不报错，提示一个Warning。如果不想等则为冲突声明错误。如果遇到的是函数定义，查询符号表，如果没有则添加一项，且设置has_body为true；如果有项目，先判断两个函数的签名是否相等，如果相等那么判断has_body，如果为true则报重复定义错误，如果为false，设置为true。

实现变量的作用域

构造SymTable的时候，构造函数：

```
class SymTable {
  SymTable() {
    FieldTable t;
    fieldTableStack.push_back(t);
  }
}
```

此时栈中有1个元素，为顶层定义，存放着全局变量。如果遇到了CompSt，那么说明遇到了大括号，则递归调用CompSt的分析之前创建新的FieldTable进栈，处理完成后再出栈。

在处理表达式中的符号的时候，从顶部开始遍历FieldTableStack，查找符号，可以保证内部的定义可以覆盖外部的定义，如果遍历了还是没找到则报错。

实现结构体的结构等价

这里就是利用了Type类型的成员函数eq的实现，所有判断Type的时候都通过eq函数，在eq中如果是结构体类型会检测结构体的字段类型是否对应相等：

```
if (this->k == Structure) {
    if (this->structure.fields.size() != other.structure.fields.size()) {
      return false;
    }
    for (size_t i = 0; i < this->structure.fields.size(); i++) {
      if (!this->structure.fields[i].type.eq(other.structure.fields[i].type)) {
        return false;
      }
    }
    return true;
}
```

如何编译项目

在项目根目录下运行

```
mkdir build && cd build
cmake ..
make
```

就可以编译项目。