编译原理实验2实验报告

功能及实现方式

我们在semantics.h/.c中实现了语义分析的接口和具体实现，其中用到的模块参见 模块介绍

- 语义分析可以分为两个大部分：
  - 定义检查：变量、数组、函数、结构体、结构体域等的定义检查
  - 类型检查：检查表达式、赋值、函数调用等的类型是否匹配
- 最终在main函数中调用的接口如下，如下列出并详细分析

```
void semantics_check() {
    /* definition checking and type checking */
    ast_walk(handle_node_preorder, handle_node_postorder);
    /* return type checking */
    ast_walk(handle_node_for_return_checking, ast_walk_action_nop);
    /* if and while condition checking */
    ast_walk(handle_node_for_condition_checking, ast_walk_action_nop);
}
```

语义分析：基本框架介绍

```
void ast_walk(ast_walk_action_t preorder_action, ast_walk_action_t postorder_action)
```

- 通过dfs遍历语法分析阶段生成的ast树，向其中传入前序和后序遍历中需要执行的函数指针
- **较为独特的设计**：
  - 提供了前序、后序遍历的函数指针，可以根据分析的需要，选择不同的遍历方式
  - 可以通过多趟dfs遍历ast树，完成不同的语义分析任务，其中的先后顺序可以保证必要的信息已处理完成

语义分析：定义检查

```
static void handle_node_preorder(ast_node_t *ast_node, int depth)
```

- 通过前序遍历来处理定义检查
- 思路如下：
  - 根据AST节点的类型，例如`AST_NODE_CompSt`, `AST_NODE_ExtDef`等，来调用相应非终结符的处理函数如`handle_compst`, `handle_extdef`等，逐层调用，最终能够处理所有的定义
  - 在处理函数中调用`production`模块提供的接口，来匹配该节点对应的产生式
  - 确定产生式之后，获得进行定义的节点，如`VarDec`, `FunDec`, `StructDec`等，调用`symtable`模块提供的接口，来进行符号表的查找与添加操作，在这个过程中完成定义检查并报出错误
- **额外内容**：实现了作用域，例如在进入CompSt时，创建一个新的符号表，在离开时，删除该符号表

语义分析：类型检查

```
static void handle_node_postorder(ast_node_t *ast_node, int depth)
static void handle_node_for_return_checking(ast_node_t *ast_node, int depth)
static void handle_node_for_condition_checking(ast_node_t *ast_node, int depth)
```

- handle_node_postorder函数通过后序遍历来处理所有Exp的类型检查、结构体field的类型检查、函数参数的类型检查
    - 使用*if (ast_node->node_type == AST_NODE_Exp)*来判断当前节点是否为Exp类型
    - 其他节点类型的处理函数类似
- handle_node_for_return_checking函数通过前序遍历来处理函数的返回值类型检查
- handle_node_for_condition_checking函数通过前序遍历来处理if和while的条件表达式类型检查

模块介绍

symtable模块(symtable.h/.c hashtable.h/.c)

- 提供了符号表（包括选做作用域）的接口：添加符号、查找符号、删除符号等。以及符号表具体数据结构的实现：拉链式哈希表。
    - 实现了多层作用域的符号表
    - 并在语义分析模块中相应处理了作用域的进入和退出

type模块(type.h/.c)

- 实现了所有非终结符的类型定义，包括基本类型、数组、函数、结构体等。并提供类型相关的接口：创建类型、比较类型、结构体添加域、函数添加参数等。
    - 通过链表实现了结构体域的存储、函数参数的存储
    - 实现了结构体域的查询

production模块(production.h/.c)

- 提供了自定义产生式的方式，并且提供产生式与ast节点的匹配接口，为语义分析提供了良好的抽象
    - 通过ast模块和tree模块提供的接口，遍历ast树，匹配产生式
    - 实现了产生式的自定义
    - 实现了产生式中表达式是否为左值的判断

编译相关

环境要求

- ubuntu 20.04
- flex 2.6.4
- bison 3.5.1
- build-essential

编译指令

```
cd Code
make parser
```

最后生成的可执行文件为`parser`，可以通过`./parser /path/to/file`运行