编译原理 实验2报告

实验人：顾馨兰 202220005　周家琛 202220019
邮箱 ： gu_xinlan@163.com
任务号：7
实验时间：2021年4月

一、实验内容

必做要求：在词法分析和语法分析程序的基础上对C--源代码进行语义分析和类型检查，并打印分析结果。应对错误类型1-17进行检查和输出。
选做要求2.1：函数除了可以定义以外，允许重复的函数声明，检查错误类型18（有函数声明，无定义）和19（函数的多次声明互相冲突，或声明与定义冲突）。

二、运行方式

按照实验指定环境，在`Code/`目录下执行`make parser`.

三、实验结果

1.语义错误识别

按照实验要求输出19种错误，部分错误导致连锁反应会有多余的报错，按实验要求可以忽略。

```
./parser ../Test/test1.c
ERROR TYPE 1 at line 4: Undefined variable "j".
./parser ../Test/test2.c
ERROR TYPE 2 at line 4: Undefined function "inc".
./parser ../Test/test3.c
ERROR TYPE 3 at line 4: Redefined variable "i".
./parser ../Test/test4.c
ERROR TYPE 4 at line 6: Redefined function "func".
./parser ../Test/test5.c
ERROR TYPE 5 at line 4: Type mismatched for assignment.
./parser ../Test/test6.c
ERROR TYPE 6 at line 4: The left-hand side of an assignment must be a variable.
./parser ../Test/test7.c
ERROR TYPE 7 at line 4: Type mismatched for operands.
./parser ../Test/test8.c
ERROR TYPE 8 at line 4: Type mismatched for return.
./parser ../Test/test9.c
ERROR TYPE 9 at line 7: Function is not applicable for arguments.
./parser ../Test/test10.c
ERROR TYPE 10 at line 4: The variable is not an array.
./parser ../Test/test11.c
ERROR TYPE 11 at line 4: The variable is not a function.
./parser ../Test/test12.c
ERROR TYPE 12 at line 4: The array index should be an integer.
./parser ../Test/test13.c
ERROR TYPE 13 at line 9: Illegal use of ".".
./parser ../Test/test14.c
ERROR TYPE 14 at line 10: Non-existent field "n".
ERROR TYPE 7 at line 10: Type mismatched for operands.
```

```
./parser ../Test/test15.c
ERROR TYPE 15 at line 3: Redefined or initialed field.
ERROR TYPE 15 at line 4: Redefined or initialed field.
./parser ../Test/test16.c
ERROR TYPE 16 at line 6: Duplicated name "Position".
./parser ../Test/test17.c
ERROR TYPE 17 at line 3: Undefined structure "Position".
./parser ../Test/test18.c
./parser ../Test/test19.c
ERROR TYPE 19 at line 8: Inconsistent declaration of function "func".
ERROR TYPE 18 at line 6: Undefined function "func".
```

四、实验思路

在语法分析完成语法树的建立后，从根节点开始遍历语法树，进行语义分析。语义分析的入口如下：

```
int main(){
      ...
      yyparse();        //完成语法树的建立
      ...
      Program(root);    //语义分析遍历入口
      check_func_def(); //最后检查所有被声明的函数都有定义
      ...
}
```

语义分析的程序位于`sym_table`模块，其中为每一个非终结符定义了一个处理程序，对语法分析树进行深度优先的遍历，在过程中进行语义错误的检查。

```
//High-level Definitions
void Program(Node *root);
void ExtDefList(Node *root);
void ExtDef(Node *root);
void ExtDecList(Node *root, Type *type);

//Specifiers
Type *Specifier(Node *root);
Type *StructSpecifier(Node *root);
char *OptTag(Node *root);
char *Tag(Node *root);
...
```

符号表、函数表、结构体表定义在`sym_table.c`中，使用散列表的方式存储。对于需要进行遍历的函数、结构体中的域名，建立链表进行访问。具体类型的设计参考实验手册完成。

```
St_Type *struct_table[0x3fff];
Sym_Type *symbol_table[0x3fff];
F_Type *function_table[0x3fff];
func_list* function_list;
FieldList* present_field;
```

五、实验分工

- 顾馨兰：编写语义分析实现（sym_table.c），编写实验报告；
- 周家琛：搭建语义分析框架（sym_table.h），编写测试样例调试修改。