编译原理 实验1报告

组长姓名：林浩然，学号：201250184，邮箱：201250184@smail.nju.edu.cn

组员姓名：邓尤亮，学号：201250035，邮箱：201250035@smail.nju.edu.cn

功能实现部分

基本内容

完成3.1.1节实验要求的全部内容。

实现细节：

（1）对于要求2.2，我们采用了散列表+栈的方法以支持嵌套作用域。尽管相比于其他方式，这种方式会占用更多内存空间，但考虑到实现上的方便，以及多占用的内存在可接受范围内，因此我们使用这种方式。

（2）对于要求2.3，我们使用了手册中推荐的类型表示相关数据结构，并编写专门的模块以进行类型等价的检查。

特色内容

（1）对于只由int、float字面量以及运算符组成的表达式，我们支持（在lab2阶段还不支持，但为之后的lab保留了接口）进行常量折叠，常量折叠的结果存放在EXP文法节点的constant结构内。节点结构如下(node.h: 23-45)：

```
typedef struct tree_node {
    // ... Unrelated code

    FieldList corresponding_field;
    struct _Type type;
    bool is_constant;

    union _Constant {
        uint32_t i;
        float f;
    } constant;
} Node;
```

在进行语义分析的同时，对于可以进行常量折叠的表达式，我们将会通过father.constant = exp1.constant op exp2.constant 的方式将子节点的常量传播到父节点。二元算术运算语义检查中的传播部分代码如下(semantic.c: 146)：

```
if (exp1->is_constant && exp2->is_constant) { // only exps that consists of literals and ops could be constant.
    if (exp1->type.u.basic == T_INT) {
        set_val(father, exp1->type, true, exp1->constant.i + exp2->constant.i, 0, NULL);
    }
    else if (exp1->type.u.basic == T_FLOAT) {
        set_val(father, exp1->type, true, 0, 0, NULL);
    }
    return;
}
```

（2）对于错误类型 9，我们不仅会正确报出手册要求内容，我们还会报出更加详细的错误信息。

如下面的分析目标代码所示：

```
int first(int x, int y, int z) {
    return x;
}

int main() {
    float xf = 0.0;
    int yi = 1;
    float zf = 1.0;
    int haha = 888;

    return first(xf, yi, zf, haha);
}
```

我们的报错更加详细，不只是报出参数有错误，还会报出第几个参数不匹配，在检查完每一个形参后如果参数数量不匹配，还会再报出参数不匹配错误：

```
Error type 9 at Line 11: Type unmatched at arg 1.
Error type 9 at Line 11: Type unmatched at arg 3.
Error type 9 at Line 11: Amount of args and params unmatched.
```

编译方式

在Code/目录下运行命令make，按照 Makefile 编译。