编译原理实验三

中间代码生成

实验思路：

实验三任务是在词法分析、语法分析和语义分析基础上将 C—源代码翻译成中间代码。

其中，在本次实现中， 中间代码的输出形式为线性双向结构，使用虚拟机小程序来测试中间代码的运行结果。

选做要求：一维数组类型变量可以作为函数参数，可以出现高维数组变量。

具体实现：

在具体实现时部分采用了实验指导上的数据结构，对于参与中间代码生成的变量，根据变量类型（比如 LABLE 数组指针 和 具体值等）：

```
37
38  //intermediate code struct
39  struct Operand_
40  {
41      enum {
42          t00, t01, t02, t03, t04, t05, t06
43      }kind;
44
45      union{
46          long int vNum;      //tmpvari,label num
47
48          char* value;    // vari
49
50
51          Operand name;   //taddr
52      }u;
53      Operand next;       //arg
54  };
55
```

对于代码，则也使用了对应的部分数据结构，根据运算符的运算数个数和类型做了区别，例如单目、双目以及特殊操作符：

```
struct InterCode_
{
    enum{
        t0, t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16
    }kind;  //mapping the INSTR CODE

    union{
        struct{         //return ,label,goto,read,write,arg, function
            Operand op;     //param
        }sigop;

        struct{
            Operand res,op1,op2;
        }binop;         //add sub mul div

        struct{…

        struct{
            Operand t1;
            Operand t2;
            char *op;
            Operand label;
        }ifgop;         //if_goto

        struct{
            Operand op;
            int size;
        }decop; //dec
    }u;
    InterCode prev,next;
};
```

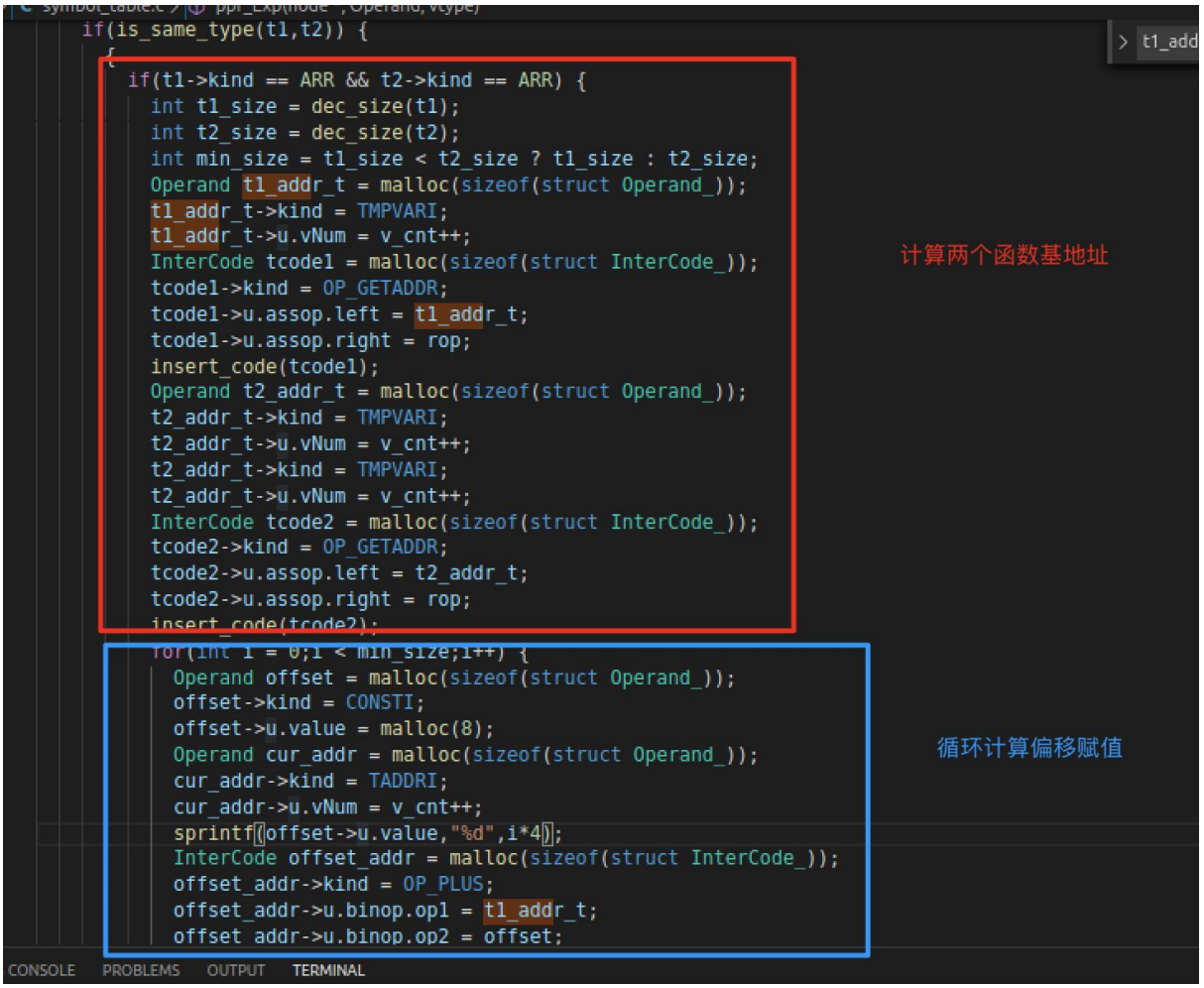
其中利用17种代码的组合能够完全实现实验要求的19种类型指令；

并封装了部分通用函数：

```
105
106
107   void insert_code(InterCode code);
108   void prt_op(Operand op, FILE* fp);
109   void prt_code(char* fname);
110
```

本次实验的选做内容为多维数组，总体来说，实验的该部分实现的并不是很好，甚至可以说是丑陋的。。具体为在原有的Exp 的SDT “ppr_SDT()”的新增了一个参数作为递归计算下，当前维度子数组的大小，而该大小是在每个数组变量由 VarDec 和 funDec 中的声明创建完后再次进行计算的，而为了保留该信息，在实验2的数组类型的信息中增加了一个域为 “esize”(意为 element size，数组元素的大小)，所以为了实现该功能不得不对原有的数据结构进行了代价比较大的改造。
实现思路就是递归到底后得到一个基地址，并加上同一层的下标*子数组大小，并递归返回计算其他下标的偏移量；最终得到完整地址；

关于数组赋值，可以认为是，两个很长的一维数组，将其中一个的所有元素填入到另一个中去，memcpy(d, s, min(sizeof(d), sizeof(s))); 于是只需要计算出两个数组的分别的声明大小，在找到基地址，然后利用for in min(sizeof(d), sizeof(s)) 循环每次将一个赋值操作给另外一个就好：



本次实验并未进行代码优化；