AST

1 符号表和 AST 结构

Common.h 中，AST 的基本结构如下：

```
typedef struct astnode{
        enum {
                KValue = 0x200,                 // numerial value:
                KName,                          // name, such as variable name
                KPrefixExp,             // prefix expression
                KInfixExp,              // infix expression
                KAssignExp,             // assignment expression
                KParenExp,              // parentheses expression
                KExpStmt,               // expression statement
                KBlock,                         // block
        } kind;  // kind of the AST node
        union {             // information of various kinds of AST node
                float  val;             // KValue: numerial value
                Symbol sym;             // KName: symbols
                Exp   exp;              // KPrefixExp,
                                        // KInfixExp,
                                        // KAssignExp,
                                        // KParenExp
                ExpStmt  estmt;                 // KExpStmt
                Block  block;           // KBlock
        };
        Loc     loc;                    // locations
} *ASTNode;
```

```
typedef struct ASTtree {

```

```
    ASTNode root;
} *ASTTree;
```

AST 节点里面主要包含两个 union，第一个 union 定义了节点的类型，主要包括数字，标识符，块，赋值语句，前缀表达式等。

第二个 union 针对第一个 union 中的每一种类型，分别定义不同的结构进行存储。例如，对数字类型的节点，定义一个 float 类型的变量来存储它。

最后一个 loc，他是一个结构体，用来保存节点对应语句的起始位置和终止位置。

2 AST 的操作函数

对于一棵 AST 树，光定义节点肯定是不够的，还需要一堆操作函数来对节点进行操作，下面是主要的 AST 操作函数。

```
ASTNode newNumber(float value);
ASTNode newName(Table ptab, char *name);
ASTNode newPrefixExp(int op, ASTNode exp);
ASTNode newParenExp(ASTNode exp);
ASTNode newInfixExp(int op, ASTNode left, ASTNode right);
ASTNode newAssignment(int op, ASTNode left, ASTNode right);
void    destroyExp(Exp *pexp);
ASTNode newExpStmt(ASTNode exp);
void    destroyExpStmt(ExpStmt *pexpstmt);
ASTNode newBlock();
void    destroyBlock(Block *pblock);
ASTTree newAST();
void    destroyAST(ASTNode *pnode);
void    dumpAST(ASTNode node);
Loc     setLoc(ASTNode node, Loc loc);
```

前五个函数针对列出的几种类型，分别创建新节点函数。如 ASTNode newNumber(float value);

是创建数字类型的节点。

再下面是这几种类型相应的删除节点的函数。

此外还有 setLoc(ASTNode node, Loc loc)，它是用来设置结点的 loc 参数，dumpAST(ASTNode node)，用来遍历并打印 AST。

在 src/symtab.c, ast.c 中有对应的函数的实现。

3 用 BISON 构造 AST 分析器

与前面构造 c0 分析器类似，AST 分析器的构造也是将 AST 的语法规则写成.y 文件，然后由 bison 自动生成.c 文件。

在 asgn2ast.y 中，有具体的实现。

asgn2ast.y 中的规则和前面的定义和函数操作等也是一一对应的，它定义了 AST 节点的不同类型，以及针对每种类型具体的操作方法。

Exp 后面的每一个语义动作即对应每种类型的操作。