技术路线

本次作业实现了一个基于LALR语法分析的FDMJ（Fudan Mini-Java）语言编译器前端，包含以下部分：

1. **语法分析器（Parser）**：使用 Bison 工具生成 LALR(1) 语法分析器，将源代码转换为抽象语法树（AST）
2. **名称解析（Name Resolution）**：构建符号表，处理变量、函数和类的作用域与访问
3. **语义分析（Semantic Analysis）**：检查类型一致性、继承关系、方法重写规则等语义约束

系统架构如下：

```
源代码 -> 词法分析器 -> 语法分析器 -> AST构建 -> 名称解析 -> 语义分析 -> 中间代码/翻译
```

关键设计决策

1. **符号表设计**：使用嵌套的哈希表结构存储类、方法和变量信息
2. **继承处理**：子类从父类继承方法和变量，支持方法重写
3. **类型检查**：支持基本类型（int）、数组类型和自定义类类型的类型检查

代码实现

语法分析器

语法分析器使用Bison工具实现，定义了FDMJ语言的文法规则：

```
PROG: MAINMETHOD CLASSDECLLIST

MAINMETHOD: PUBLIC INT MAIN '(' ')' '{' VARDECLLIST STMLIST '}'

VARDECLLIST: /* empty */
           | VARDECL VARDECLLIST

VARDECL: CLASS ID ID ';'
       | INT ID ';'
       | INT ID '=' CONST ';'
       | INT '[' ']' ID ';'
       | INT '[' ']' ID '=' '{' CONSTLIST '}' ';'
       | INT '[' NONNEGATIVEINT ']' ID ';'
       | INT '[' NONNEGATIVEINT ']' ID '=' '{' CONSTLIST '}' ';'

CONST: NONNEGATIVEINT
     | MINUS NONNEGATIVEINT

CONSTLIST: /* empty */
         | CONST CONSTREST

CONSTREST: /* empty */
         | ',' CONST CONSTREST

STMLIST: /* empty */
       | STM STMLIST
```

```
STM: '{' STMLIST '}'
   | IF '(' EXP ')' STM %prec IFX
   | IF '(' EXP ')' STM ELSE STM
   | WHILE '(' EXP ')' STM
   | WHILE '(' EXP ')' ';'
   | EXP '=' EXP ';'
   | EXP '.' ID '(' EXPLIST ')' ';'
   | CONTINUE ';'
   | BREAK ';'
   | RETURN EXP ';'
   | PUTINT '(' EXP ')' ';'
   | PUTCH '(' EXP ')' ';'
   | PUTARRAY '(' EXP ',' EXP ')' ';'
   | STARTTIME '(' ')' ';'
   | STOPTIME '(' ')' ';'

EXP: NONNEGATIVEINT
   | TRUE
   | FALSE
   | LENGTH '(' EXP ')'
   | GETINT '(' ')'
   | GETCH '(' ')'
   | GETARRAY '(' EXP ')'
   | ID
   | THIS
   | EXP ADD EXP
   | EXP MINUS EXP
   | EXP TIMES EXP
   | EXP DIVIDE EXP
   | EXP EQ EXP
   | EXP NE EXP
   | EXP LT EXP
   | EXP LE EXP
   | EXP GT EXP
   | EXP GE EXP
   | EXP AND EXP
   | EXP OR EXP
   | NOT EXP
   | MINUS EXP %prec UMINUS
   | '(' EXP ')'
   | '(' '{' STMLIST '}' EXP ')'
   | EXP '.' ID
   | EXP '.' ID '(' EXPLIST ')'
   | EXP '[' EXP ']'

EXPLIST: /* empty */
       | EXP EXPREST

EXPREST: /* empty */
       | ',' EXP EXPREST

CLASSDECLLIST: /* empty */
            | CLASSDECL CLASSDECLLIST

CLASSDECL: PUBLIC CLASS ID '{' VARDECLLIST METHODDECLLIST '}'
         | PUBLIC CLASS ID EXTENDS ID '{' VARDECLLIST METHODDECLLIST '}'

METHODDECLLIST: /* empty */
```

```
              | METHODDECL METHODDECLLIST

METHODDECL: PUBLIC TYPE ID '(' FORMALLIST ')' '{' VARDECLLIST STMLIST '}'

TYPE: CLASS ID
    | INT
    | INT '[' ']'

FORMALLIST: /* empty */
          | TYPE ID FORMALREST

FORMALREST: /* empty */
          | ',' TYPE ID FORMALREST

ID: IDENTIFIER
```

文法规则定义了程序的结构，包括主方法、类声明、方法声明等，以及表达式和语句的语法。

特别注意处理了运算符的优先级和结合性：

```
// 优先级从低到高排列
%left ','                       // 逗号（用于 ExpList 和 FormalList）
%right '='                      // 赋值运算符
%left OR                        // 逻辑或 ||
%left AND                       // 逻辑与 &&
%left EQ NE                     // 相等性运算符 == !=
%left LT LE GT GE               // 关系运算符 < <= > >=
%left ADD MINUS                 // 加减运算符 + -
%left TIMES DIVIDE              // 乘除运算符 * /
%right UMINUS                   // 一元减号 -（优先级高于二元加减，但低于乘除）
%right NOT                      // 一元逻辑非 !
%left '.'                       // 成员访问 .
%left '[' ']'                   // 数组索引 []
%left '(' ')'                   // 括号（最高优先级）

%nonassoc IFX       // 无 else 的 if 语句
%nonassoc ELSE      // else 分支
```

名称解析

`setnamemaps.cc` 实现了名称解析，构建符号表并处理标识符绑定。主要功能包括：

1. **类层次结构处理**：

```
if (node->eid != nullptr) {
    if (!name_maps->add_class_hiearchy(node->id->id, node->eid->id)) {
        cerr << node->eid->getPos()->print() << endl;
        cerr << "- Class inheritance error: " << node->id->id << " extends " <<
node->eid->id << endl;
        exit(1);
    }
}
```

2. **方法重写检查**：

```
if (!is_same_type(t1, t2)) {
    cerr << node->id->getPos()->print() << endl;
    cerr << "- Override method return type mismatch: " << node->id->id << "->"
<< m << endl;
    exit(1);
}
```

语义分析

`semantanlyzer.cc`实现了语义分析，检查类型一致性和程序的语义约束：

1. **类型检查**：

```
bool AST_Semant_Visitor::is_assignable(AST_Semant* left, AST_Semant* right) {
    // 检查类型匹配
    if (left->get_type() != right->get_type())
        return false;

    // 特殊处理类继承关系
    if (left->get_type() == TypeKind::CLASS) {
        // 检查是否为子类关系
    }

    // 检查数组维度
    if (left->get_type() == TypeKind::ARRAY) {
        // 检查数组维度匹配
    }
}
```

2. **控制流检查**：

```
void AST_Semant_Visitor::visit(Continue* node) {
    // 检查在while里
    if (!is_in_while) {
        cerr << node->getPos()->print() << endl;
        cerr << "- Continue statement is not inside a while loop" << endl;
        exit(1);
    }
}
```

测试脚本

实现了一个自动化测试脚本`parser_test.bash`，用于编译程序并对测试用例进行批量测试，核心代码如下：

```
for fmj_file in ../test/*.fmj; do
    if [ -f "$fmj_file" ]; then
        filename="${fmj_file%.fmj}"
        base_name=$(basename "$fmj_file" .fmj)
        echo "Testing file: $base_name"

        # 运行测试程序
        ./tools/main/main "$filename"
    fi
done
```

实验效果

```
(base) keats@OMEN-Yanxu:~/FudanCompilerH2025/HW3$ bash parser_test.bash
Output directory exists. Cleaning it...
-- Configuring done (0.0s)
-- Generating done (0.0s)
-- Build files have been written to: /home/keats/FudanCompilerH2025/HW3/build
[ 30%] Built target frontend
[ 38%] Building CXX object lib/ast/CMakeFiles/ast.dir/semantanlyzer.cc.o
[ 61%] Built target ast
[ 76%] Built target util
[ 84%] Built target vendor_xml
[ 92%] Linking CXX executable main
[100%] Built target main
Build successful. Running parser tests...
Testing file: bubblesort
------Parsing fmj source file: ../test/bubblesort.fmj------------
Convert AST  to XML...
Writing AST to file: ../test/bubblesort.2.ast
Read AST from file: ../test/bubblesort.2.ast
Converting XML to AST...
Semantic analysis...
--Making Name Maps...
--Analyzing Semantics...
Convert AST to XML with Semantic Info...
---------------------------------------
Testing file: example
------Parsing fmj source file: ../test/example.fmj------------
Convert AST  to XML...
Writing AST to file: ../test/example.2.ast
Read AST from file: ../test/example.2.ast
Converting XML to AST...
Semantic analysis...
--Making Name Maps...
--Analyzing Semantics...
Position(sline: 14, scolumn: 15, eline: 14, ecolumn: 38)
- Class method call parameter count does not match
---------------------------------------
Testing file: fibonacci
------Parsing fmj source file: ../test/fibonacci.fmj------------
Convert AST  to XML...
Writing AST to file: ../test/fibonacci.2.ast
Read AST from file: ../test/fibonacci.2.ast
Converting XML to AST...
Semantic analysis...
--Making Name Maps...
--Analyzing Semantics...
Convert AST to XML with Semantic Info...
---------------------------------------
```

提交记录

```
(base) keats@OMEN-Yanxu:~/FudanCompilerH2025/HW3$ git log --graph
* commit f893da3d1073f309f3e2927b3009eaeb9ba2f6db (HEAD -> main)
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 27 23:05:57 2025 +0800
|
|     HW3 pass (better to upgrade
|
* commit f5bafb5c2f8b6a85990323fd8a950d0648be1b44
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 27 22:48:35 2025 +0800
|
|     finish write work, need to debug
|
* commit b7199909ac47e9b2ebf3cd4e348b836b461f5677
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 27 16:26:49 2025 +0800
|
|     work on type checking
|
* commit aaec8db2330a0e5f59cdef40c78d4e9827b6dabf
| Author: xht03 <1620318777@qq.com>
| Date:   Wed Mar 26 21:10:04 2025 +0800
|
|     update on parser.yy
|
* commit 955a691205f5003f4fd7c89b135a8f028bbc90d6 (origin/main)
| Author: xht03 <1620318777@qq.com>
| Date:   Sat Mar 22 20:29:14 2025 +0800
|
|     pull HW3 from gitee
|
* commit f0f12615c6e2eaf28059d91840b4dab63ef785e1
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 13 19:17:59 2025 +0800
|
|     add report
|
* commit 8043a4d41c438546d8d83f0f5098ead455d5caab
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 13 16:55:36 2025 +0800
|
|     lab2 done!
|
* commit 87b87dbd3702d21591ffa99b8a060dae457f5588
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Mar 13 10:02:08 2025 +0800
|
|     upgrade 3.3d
|
* commit a1a78c9f25a23239eeb180361709fc39163ebe8c
| Author: xht03 <1620318777@qq.com>
| Date:   Wed Mar 12 21:32:29 2025 +0800
|
|     fix 3.3d
```