实验任务4.1： SysY语言的语法和语义规范理解 实验报告

实验目的

本实验旨在通过分析SysY语言的语法和语义规范，加深对编译原理中语法分析和语义分析部分的理解。通过编写测试程序和分析其语法成分，学生将能够更好地理解语言规范的实现和应用。

SysY语言核心语法成分分析

SysY是C语言的子集扩展，专为编译器设计竞赛使用，支持基本类型、数组、多种语句及表达式，采用EBNF表示。SysY的程序结构包括全局变量/常量声明、函数定义，并支持算术、关系和逻辑运算。

编译单元 `CompUnit`

作用：表示SysY程序的顶层结构，包含声明和函数定义。

文法规则：`CompUnit → [ CompUnit ] ( Decl | FuncDef )`

限定条件：必须包含且仅包含一个无参数、返回 `int` 类型的 `main` 函数；`Decl` 和 `FuncDef` 中标识符不能重名。

组合情况：`Decl` 与 `FuncDef` 可交替出现。

示例句子

```
const int a = 5;
int b;
int main() {
    return 0;
}
```

声明 `Decl`

作用：声明用于进行常量或变量的声明及进行初始化，其中，声明过程由 `ConstDecl` 和 `VarDecl` 完成，具体的定义过程由 `ConstDef` / `VarDef` 定义完成.

文法规则：`Decl → ConstDecl | VarDecl`

限定条件：

- 常量需显式初始化
- 变量声明可选初始化

组合情况：可出现在 `CompUnit` 或函数内部的 `Block` 中。

示例句子

```
const int maxSize = 100;
float x = 3.14;
```

常量声明 `ConstDecl`

文法规则：

```
ConstDecl → 'const' BType ConstDef { ',' ConstDef } ';'
ConstDef → Ident { '[' ConstExp ']' } '=' ConstInitVal
```

作用：声明不可修改的常量。

限定条件：

- 初值必须是常量表达式；
- 支持数组形式（需固定大小）。

示例句子：

```
const int a = 10, b[3] = {1, 2, 3};
```

变量声明 `VarDecl`

文法规则：

```
VarDecl → BType VarDef { ',' VarDef } ';'
VarDef → Ident { '[' ConstExp ']' } | Ident { '[' ConstExp ']' } '=' InitVal
```

作用：声明变量并可选通过变量定义进行初始化。

示例句子：

```
int x = 5, y[2][3];
```

函数定义 `FuncDef`

文法规则：`FuncDef → FuncType Ident '(' [FuncFParams] ')' Block`

作用：用于定义一个函数，其中函数体的部分用 `Block` 进行进一步的表示

限定条件：`FuncType` 可为 `void`、`int`、`float`

组合情况：参数通过 `FuncFParams` 定义，函数体由 `Block` 表示。

示例句子：

```
int add(int a, int b) {
    return a + b;
}
```

表达式 `Exp`

定义和功能

表达式是 SysY 语言中的基本计算单位，包含算术、关系、逻辑等运算，计算后返回 `int` 或 `float` 型的值。表达式在赋值语句、条件语句、循环语句等上下文中广泛使用。其中，表达式中的运算符优先级与C语言一致.

语法规则

表达式 `Exp`

```
Exp → AddExp
```

注：SysY 的表达式是 `int` 或 `float` 类型。

加减表达式 `AddExp`

```
AddExp → MulExp | AddExp ('+' | '−') MulExp
```

乘除模表达式 `MulExp`

```
MulExp → UnaryExp | MulExp ('*' | '/' | '%') UnaryExp
```

一元表达式 `UnaryExp`

```
UnaryExp → PrimaryExp | Ident '(' [FuncRParams] ')' | UnaryOp UnaryExp
```

注：`UnaryOp` 是单目运算符，包括 `+`、`-` 和 `!`，其中 `!` 仅在条件表达式中使用。

基本表达式 `PrimaryExp`

```
PrimaryExp → '(' Exp ')' | LVal | Number
```

语法成分分析

语法成分	作用	限定条件	组合情况	示例句子
`Exp`	计算并返回值	`int` 或 `float` 类型	可嵌套在条件表达式、赋值语句等中	`a + b * 3`
`AddExp`	进行加法或减法运算	左结合，`+` 和 `-` 优先级低于 `*` 和 `/`	由 `MulExp` 组成	`1 + 2 - 3`
`MulExp`	进行乘法、除法或取模运算	左结合，`*`、`/` 和 `%` 优先级高	可组合为更复杂的表达式	`a * (b + 1)`
`UnaryExp`	处理单目运算或函数调用	函数调用需声明函数原型	嵌套在更复杂表达式中	`-a`，`!flag`，`func(a)`
`PrimaryExp`	表达式最小单元	`LVal`、`Number` 或括号内合法表达式	可组合到更复杂表达式中	`(a + 1)`，`10`，`arr[2]`

语句 `Stmt`

定义和功能

语句是 SysY 语言中程序执行的基本单位，用于变量赋值、流程控制或函数调用等，直接影响程序的逻辑和控制流。

语法规则

语句

```
Stmt → LVal '=' Exp ';' | [Exp] ';' | Block
       | 'if' '(' Cond ')' Stmt [ 'else' Stmt ]
       | 'while' '(' Cond ')' Stmt
       | 'break' ';' | 'continue' ';'
       | 'return' [Exp] ';'
```

条件表达式

```
Cond → LOrExp
```

语法成分分析

语法成分	作用	限定条件	组合情况	示例句子
赋值语句	为变量赋值	`LVal` 必须可被赋值	嵌套在 Block、循环等结构中	`a = b + 1;`
表达式语句	执行表达式计算	可为空表达式	独立作为语句，末尾加分号	`func_call();`，`;`
语句块（Block）	表示代码块	包含变量声明或语句	嵌套在控制结构中	`{ int a; a = 1; return; }`
条件语句（if-else）	实现分支控制	条件必须是 `Cond`	嵌套复杂的条件或语句块	`if (a > 0) return; else a = -a;`
循环语句 (while)	实现循环	条件必须是 `Cond`	包含 Block 或其他语句	`while (a < 10) a = a + 1;`
跳转语句 (break/continue)	改变循环执行	必须出现在循环中	结合 `while` 使用	`if (a == 0) break;`
返回语句 (return)	返回函数结果	`Exp` 类型与函数返回类型一致	函数中结束执行	`return a + b;`