Calculator-Tools

1 计算器说明

计算器接受四则运算表达式为输入。如果表达式语法正确，则输出计算结果，**否则报错，指出错误位置及原因。**

1.1 基础功能

1. 支持浮点数和整数两种类型，浮点数可以转换成整数；~~整数不可以转换成浮点数。~~
2. 每个语句需要以“；”结束，以“.”表示全部输入的结束。
3. 变量需要先声明再使用。
4. 变量名可以是由数字和字母组成，但首字符必须是字母。
5. 每个表达式中使用的变量需要在之前已经有赋值。
6. 输出语句使用write(a)，输出并换行，其中a为int类型或者float类型的变量名。

1.2 补充功能

1. 支持浮点数与整数的相互转换，发生类型不匹配时的转换规则如下：
    - 进行赋值操作时，将右值的类型转换为左值的类型；
    - 进行数值运算时，将整型变量转换为浮点型变量。
2. 支持在变量声明时同时进行初始化，如`int a = 2;`。
3. 输出语句write()支持输出表达式，如`write(a * 2 + 3);`。

1.3 错误处理

当发现某一行出现错误时，报告错误的位置及原因，并终止程序运行。

2 文法设计

2.1 语法

program → decls stmts .

decls → decls decl | ε

decl → **type id** ; | **type id** = expr ;

stmts → stmts stmt | ε

stmt → **id** = expr ; | **write** (expr) ;

expr → expr + term | expr - term | term

term → term * unary | term / unary | unary

unary → - unary | factor

factor → (expr) | **number** | **id**

2.2 词法

digit → [0-9]

digits → digit+

number → digits (. digits)?

letter → [A-Za-z]

id → letter (letter|digit)*

type → int|float

write → write

3 程序设计实现

3.1 源文件功能说明

- main.c 主函数，负责进行文件读写等
- Lexer.c 词法分析器
- Lexer.h Lexer.c的头文件
- Parser.c 语法分析器
- Parser.h Parser.c的头文件
- Functions.h 字符操作函数等定义
- CMakeLists.txt cmake脚本
- build/ cmake生成的工程文件目录
- test/ 测试文件目录

3.2 词法分析

在词法分析阶段首先除去空白和换行符。随后依次尝试提取数字，以字母开头的字符串，单字符。最后返回一个Token，若识别到标识符，还要将其加入到变量表中。

3.3 语法分析

设计**语法制导翻译方案**如下：

program → decls stmts .

decls → decls decl | ε

decl → **type id** ; {id加入变量表}

| **type id** = expr ; {id加入变量表， assign(id, value_expr)}

stmts → stmts stmt | ε

stmt → **id** = expr ; {assign(id, value_expr)}

| **write** (expr) ; {print(value_expr)}

expr → expr + term {return value_expr + value_term}

| expr - term {return value_expr - value_term}

| term {return value_term}

term → term * unary {return value_term * value_unary}

| term / unary {return value_term / value_unary}

| unary {return value_unary}

unary → - unary {return -1 * value_unary}

| factor {return value_factor}

factor → (expr) {return value_expr}

| **number** {return value_number}

| **id** {return value_id}

在语法分析阶段使用**自顶向下分析方法**，入口函数为Parser()，并使用以下转换式消除左递归：

A → Aα | Aβ | γ

转换为

A → γR

R → αR | βR | ε

5 测试样例

通过test_batch.py脚本调用程序进行测试，测试样例与结果输出全部位于test目录下，测试样例储存在test*.txt，输出结果储存在out*.txt。

5.1 test1

```
Input 1:
float a;
int b;
a = (10.44*356+1.28) / 2 + 1024 * 1.6;
b = a * 2 - a/2;
write(b);
write(a);.

Output 1:
5246
3497.360000
```

5.2 test2

```
Input 2:
float a;
int b;
a = (10.44 * 356 + 1.28) / 2 + 1024 * 1.6;
b = a * 2 - c/2;
write(b);.

Output 2:
ERROR(line 4): identifier 'c' is undeclared.
```

5.3 test3

```
Input 3:
float qwe123;
int ppp;
qwe123 = 6 + 2.33 * 1.11 - (26817 * 3.12345 + 4);
write(qwe123);
ppp = qwe123 + 5;
write(ppp);
ppp = 2 * 33.001;
write(ppp);
write(123);
qwe123 = 0.0;
ppp = 2 / qwe123;
write(ppp);.

Output 3:
-83756.972350
-83751
66
123
ERROR(line 11): the divisor is zero.
```

5.4 test4

```
Input 4:
float aaa123 = 3.14159;
int bbb123 = -1000 + -2 * aaa123;
float ccc123;
int undef;
ccc123 = aaa123 / bbb123;
write(aaa123);
write(bbb123);
write(ccc123);
write(undef);.

Output 4:
3.141590
-1006
-0.003123
ERROR(line 9): identifier 'undef' is unassigned.
```

5.5 test5

```
Input 5:
float xyz123 = -98.12345;
int q1q2q3;
q1q2q3 = xyz123 * (--(-1234)) + 1;
write(xyz123 * 2);
write(q1q2q3);.

Output 5:
-196.246900
121085
```