Minimal BASIC 设计文档

1. Minimal BASIC 简介

Minimal BASIC 是一个精简版的 BASIC 语言解释器。它支持的 BASIC 语句有 REM、LET、PRINT、INPUT、GOTO、IF、END。它支持的控制命令有 RUN、LIST、CLEAR、HELP、QUIT。

Minimal BASIC 仅支持 ASCII 字符，对大小写不敏感。

Minimal BASIC 支持的表达式仅能包含 32 位有符号整数数据类型的变量与常数，以及该数据类型上的如下运算：加法（+）、减法（-）、乘法（*）、整数除法（/）、乘方（**）、单目加（+）、单目减（-）。被 0 除、0 的 0 次方会引发报错。运算中的数据溢出行为无明确规定。

Minimal BASIC 要求变量名仅能包含英文字母、数字、下划线，不能为空，第一个字符不能是数字，不能与前述 BASIC 语句、控制命令中的关键字相同。

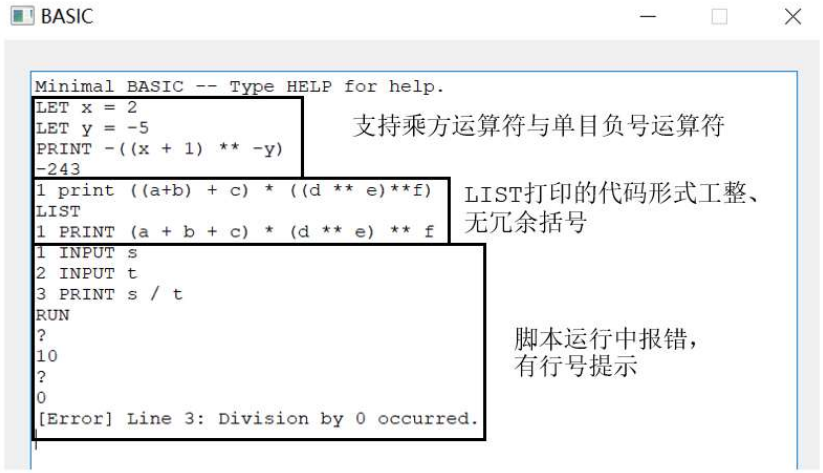

图 1 Minimal BASIC 部分功能

2. 模块划分

表 1 模块划分

头文件	功能
context.h	Context 类用于维护 Minimal BASIC 运行过程中定义的变量以及它们的值。
expression.h	Expression 类及其子类实现了表达式树。
tokenizer.h	Tokenizer::tokenize 函数能将用户输入的表达式字符串拆分成一个记号列表。
parser.h	Parser::parse 函数能解析用户输入的表达式字符串，并返回一棵表达式树。在用户输入的表达式无法解析时，Parser::parse 会抛出异常。

statement.h	Statement 类及其子类实现了 Minimal BASIC 所支持的 BASIC 语句。
program.h	Program 类实现了 Minimal BASIC 脚本。
error.h	Error 类及其子类提供了异常处理机制所需要的异常类。
console.h	Console 类实现了控制台。
basic_window.h	BasicWindow 类实现了 Minimal BASIC 的主窗口。

3. 模块实现思路

3.1. Statement 类与 Program 类

Statement 类的七个子类 RemStmt、LetStmt、PrintStmt、InputStmt、EndStmt、GotoStmt、IfStmt 分别实现了 REM、LET、PRINT、INPUT、END、GOTO、IF 七种 BASIC 语句。Statement 类的纯虚函数 Statement::exec 实现了语句的执行。

Program 类，是一个 Statement 对象列表。Program 类中 QMap<int, Statement *>类型的私有成员 stmts，记录了脚本中各个语句的编号与内容。脚本的运行，总体上可看作按编号从小到大的顺序调用 stmts 中各个 Statement 对象的 exec 函数。

Program 类的私有成员 pc、state 在脚本运行过程中起表征脚本运行状态的作用。pc 的类型是 QMap<int, Statement *>::const_iterator，表征了当前正在执行的语句。state 的类型是 enum State，在必要时起表征脚本运行状态的作用，其可能取值如表 2 所示。

表 2 Program 类私有成员 state 的可能取值

取值	含义
NONE	无特殊状态
INPUT	等待输入
OUTPUT	等待输出
AFTER_INPUT	等待处理输入
END	脚本运行结束

Program 类的私有成员 input、output 均为 QString 类型，input 用于在脚本运行过程中接收输入，output 用于在脚本运行过程中给出输出。

Statement::exec 的函数原型为 void Statement::exec(Program *prog, Context *context)，prog 是正在运行的脚本，context 是脚本运行所基于的变量环境。表 3 简述了七种 BASIC 语句的 exec 函数的实现。

表 3 七种 BASIC 语句的 exec 函数的实现简述

函数	实现
RemStmt::exec	1. prog->pc 加一。
LetStmt::exec	1. 在 context 中设置变量。 2. prog->pc 加一。
PrintStmt::exec	1. 表达式求值（可能抛出异常）。 2. 设置 prog->output。 3. 设置 prog->state 为 OUTPUT。

	4．prog->pc 加一。
InputStmt::exec	若 prog->state 不是 AFTER_INPUT： 1．设置 prog->state 为 INPUT。 若 prog->state 是 AFTER_INPUT： 1．检查输入合法性，发现输入不合法则抛出异常。 2．在 context 中设置变量。 3．设置 prog->state 为 NONE。 4．prog->pc 加一。
EndStmt::exec	1．设置 prog->pc 为 prog->stmts.cend()。
GotoStmt::exec	1．若跳转目标在 prog->stmts 中不存在，则抛出异常。 2．设置 prog->pc 为跳转目标。
IfStmt::exec	首先，表达式求值（可能抛出异常）。 若跳转条件成立： 1．若跳转目标在 prog->stmts 中不存在，则抛出异常。 2．设置 prog->pc 为跳转目标。 若跳转条件不成立： 1．prog->pc 加一。

当需要运行一个脚本时，首先调用 Program::run 函数，依次执行各 Statement 对象的 exec 函数，直至发生以下情况之一：等待输入、等待输出、exec 函数抛出异常、脚本运行结束。若 Program::run 结束后，脚本等待输出，则在用户界面上输出 output 中的内容，设置 state 为 NONE 以表征输出完成，然后再次调用 Program::run 函数。若 Program::run 结束后，脚本等待输入，则等待用户输入内容后，将用户输入的内容放入 input，设置 state 为 AFTER_INPUT 以表征输入完成，再次调用 Program::run 函数。如此不断地运行 Program::run，输入或输出，运行 Program::run，输入或输出，……，直至 exec 函数抛出异常或脚本运行结束，就是 Minimal BASIC 的脚本运行机制。

Minimal BASIC 的脚本运行机制不依赖于 Qt 机制，这使得 Minimal BASIC 可以被便捷地移植到其他平台上。