# Minimal BASIC设计文档

### 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。

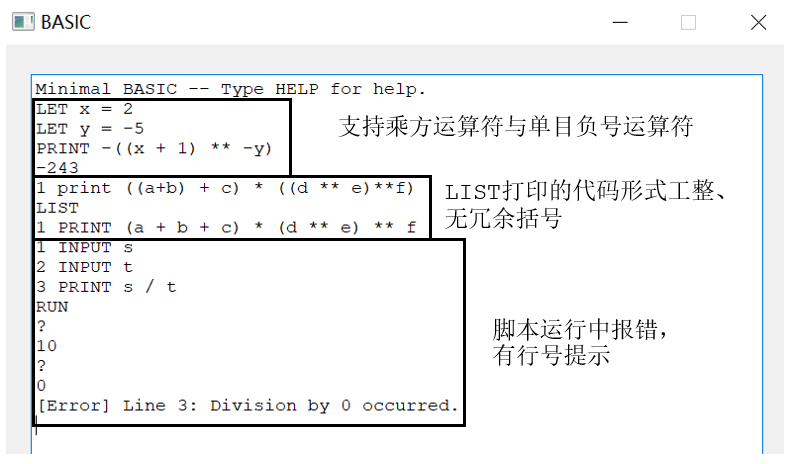


图 1 Minimal BASIC部分功能

### 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。 |

### 模块实现思路

##### 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。