# Computational-Graph

## 运行环境

Linux， Windows 平台皆可，推荐 Linux

软件包依赖 `CMake`，要求版本 `3.10` 以上

## 测试代码编译命令

### Linux 环境

在 `.` 目录下运行

```
$ cmake .
$ make
$ ./main
```

### Windows 环境

请使用具体的 `CMake` 软件构建工程

构建成功后可执行文件为 `main.exe`

## 测试程序运行方法

### 概述

在 Linux 下运行 `main` 或 Windows 下运行 `main.exe` 即可运行程序

程序由键盘输入，可以利用 `main < INPUT_FILENAME > OUTPUT_FILENAME` 对输入输出进行重定向

测试程序的错误信息通过 `std::cout` 输出，因此会被一起重定向

输入需包含三部分：独立节点建立、依赖节点建立、命令执行

如果出现输入格式错误，可能会导致输出异常结果，甚至程序异常终止

### 独立节点建立

首先输入一个整数 $n$，表示要建立 $n$ 个独立节点

接下来 $n$ 行，每行描述一个独立节点，格式为

`NODENAME NODETYPE ...`

`NODENAME` 是一个不含空格的字符串，表示新建节点的名称，可以与之前已建立的节点重名，但会导致该节点名称与原先的节点脱离关系

`NODETYPE` 是一个大写字母 `P`，`C` 或 `V`，分别表示建立的节点类型为 **占位节点**， **常量节点** 或 **变量节点**

如果建立的节点类型为 `P`，则 `...` 为空

如果建立的节点类型为 `C` 或 `V`，则 `...` 是一个实数，表示节点的初始值

# 依赖节点建立

首先输入一个整数 $n$，表示要建立 $n$ 个依赖节点

接下来 $n$ 行，每行描述一个依赖节点

## 建立双目运算节点

格式为

`NODENAME = OPERAND1 OPERATOR OPERAND2`

`NODENAME` 是一个不含空格的字符串，表示新建节点的名称，不能与之前已建立的节点重复，否则输出错误信息 `ERROR: NODENAME already exist`，并强行终止程序

`OPERAND1` 与 `OPERAND2` 是两个已建立的节点名称

`OPERATOR` 是一个运算符字符串 `+`，`-`，`*`，`/`，`>`，`<`，`>=`，`<=` 或 `==` 表示对应的运算

## 建立高级运算节点

格式为

`NODENAME = FUNCTION OPERAND`

`NODENAME` 是一个不含空格的字符串，表示新建节点的名称，不能与之前已建立的节点重复，否则输出错误信息 `ERROR: NODENAME already exist`，并强行终止程序

`FUNCTION` 为 `EXP`，`LOG`，`SIN`，`TANH` 或 `SIG` 分别表示对应的高级计算节点

`OPERAND` 是一个已建立的节点名称

## 建立调试输出节点

格式为

`NODENAME = PRINT OPERAND`

`NODENAME` 是一个不含空格的字符串，表示新建节点的名称，不能与之前已建立的节点重复，否则输出错误信息 `ERROR: NODENAME already exist`，并强行终止程序

`OPERAND` 是一个已建立的节点名称

调试输出节点的作用为，执行每条命令时，对被观察节点的值进行第一次求值完成后，会输出被观察节点的值

## 建立条件运算节点

格式为

`NODENAME = COND OPERAND0 OPERAND1 OPERAND2`

`NODENAME` 是一个不含空格的字符串，表示新建节点的名称，不能与之前已建立的节点重复，否则输出错误信息 `ERROR: NODENAME already exist` ，并强行终止程序

`OPERAND0`，`OPERAND1` 与 `OPERAND2` 是三个已建立的节点名称

条件运算节点的作用为，在 `OPERAND0` 的值大于 0 时，返回 `OPERAND1` 的值，否则返回 `OPERAND2` 的值

# 命令执行

## 求值命令

格式为

`EVAL NODENAME K OP1 V1 OP2 V2 ... OPK VK`

`NODENAME` 是一个已建立节点的名称，如果未找到，则输出错误信息 `ERROR: NodeName NODENAME not found` ，并强行终止程序

`K` 是一个整数，表示要赋值的占位符数量

`OPi (i=1,2,...,K)` 为一个占位节点的名称； `Vi (i=1,2,...,K)` 为一个实数

表示，仅在这次计算中，将 `OPi` 赋值为 `Vi`

如果在运算过程中出现操作元的值不满足运算符的要求，会输出错误信息并终止此次计算，但不会影响之后的命令

## 变量赋值为历史答案

格式为

`SETANSWER NODENAME K`

`NODENAME` 是一个已建立节点的名称，如果未找到，则输出错误信息 `ERROR: NodeName NODENAME not found` ，并强行终止程序

`K` 是一个整数，表示将第 $K$ 次命令的答案赋值给 `NODENAME`

保证该次操作为求值命令

## 变量赋值为常量

格式为

`SETCONSTANT NODENAME V`

`NODENAME` 是一个已建立节点的名称，如果未找到，则输出错误信息 `ERROR: NodeName NODENAME not found` ，并强行终止程序

`V` 是一个实数，表示将 $V$ 赋值给 `NODENAME`

# 错误信息速查

## `ERROR: Failed to build NODENAME`

在构建 `NODENAME` 节点时输入格式错误

### `ERROR: NodeName NODENAME not found`

在需要输入已存在节点的名称时，输入的名称不存在

### `ERROR: Placeholder missing`

在计算时有被依赖的占位符未赋值

### `ERROR: Variable missing`

在计算时有被依赖的变量未赋值/已被清除

### `ERROR: Division by zero`

除法运算时除数为 0

### `ERROR: LOG operator's input must be positive`

对数运算时真数不为正数

## 如果你想用我的库

请按如下格式编写 `CMakeLists.txt`

```
CMAKE_MINIMUM_REQUIRED(VERSION 3.10)
PROJECT(YOUR_PROJECT) #YOUR_PROJECT_NAME是你的工程名

ADD_SUBDIRECTORY(lib)
ADD_SUBDIRECTORY(basic_calc_pack)
ADD_SUBDIRECTORY(advanced_calc_pack)
ADD_SUBDIRECTORY(compare_calc_pack)

ADD_EXECUTABLE(YOUR_EXECUTABLE ...) #YOUR_EXECUTABLE是你想编译的可执行文件名；...为该可执行文件依赖的你编写的所有源文件
TARGET_LINK_LIBRARIES(YOUR_EXECUTABLE Lib Basic_Calc_Pack Advanced_Calc_Pack Compare_Calc_Pack)
#如果你要生成多个可执行文件，请多次使用以上两行语句
```