计算图实现介绍

在主文件main.cpp中按照题目格式要求解析输入流，调用API创建常量、变量、占位符等，建立计算图，最后求值并输出。在此并不做任何实质性的计算工作，仅负责与题目交互，计算工作交由计算图库完成。

以下计算图库的内容除了arithmetics.h/cpp外全部封装在cgraph命名空间中。

计算图中所有的节点均为虚基类Node的派生类对象，Node类在Node.h/cpp中声明并实现。Node类对象中，value代表该节点的临时值，version表示这一临时值的版本；visited表示该节点是否已经在当前轮计算中被访问过。为了支持计算，Node类需要实现三个虚函数：getPreq、getAllPreq应该返回计算这个节点需要先行计算的节点，eval用于更新value和version。getPreq和getAllPreq的区别在于， getAllPreq用于得到计算该节点需要的所有节点；getPreq用于得到计算该节点的却未被计算的节点 。由于value和version仅应在eval函数中修改，这两个变量的访问权限均为protected，需要使用getValue和getVersion得到对象相应的value和version。

Node::version和Node::visited为Symbol类的对象。Symbol类的对象起到了唯一标识符的作用，用普通构造函数构造的对象和已有对象均不相等，而通过拷贝构造函数或赋值运算符复制的对象两两相等。Symbol类在Symbol.h/cpp中实现。

Expression.h/cpp中完成的是Expression类。Expression类封装了一个智能指针，指向Node或其派生类对象，外部使用时并不需要关心内存资源分配及回收问题；Expression的成员函数eval实现了遍历该节点的后续节点并求值的操作。

大部分运算的调用为运算符重载的形式，定义为接收Expression类对象、返回Expression类对象。需要注意这里并没有真的去对数值进行运算，而只是新建了一个代表该运算的节点。每种运算都从Node类派生出新的节点类，在arithmetics.h/cpp中实现。而在相应的类中，getAllPreg与getPreq的功能与Node类中保持一致；而在计算值（eval函数中）时，若其节点的前驱节点值并未变化，那么不需要重新计算，否则计算并更新版本号。若计算时出现了定义域错误、分母为0、占位符未定义等情况，则抛出在Error.h/cpp中创建的异常类的对象。

在sinks.h/cpp中声明并定义了三个不同的终止节点类——Placeholder类、Constant类和Variable类。对于Variable类，只要保留了该类的对象，就可以对变量的值进行操作。这三个类的对象均可以转为Expression对象以支持在Expression上定义好的加减乘除等运算。因为Placeholder类型的对象需要在求值之前赋值，所以在全局定义了一个eval函数，该函数接收一个Expression和一个Placeholder映射到浮点数值的std::map，返回代入这些值时Expression的值。

Graph.h/cpp中声明并实现了Graph类。Graph类对象维护了变量名到节点的映射，内部使用std::map实现。

最后，使用cgraph.h可以一次性将所有头文件引入。

**在可扩展性上的一些优点**

为了防止节点依赖链过长，递归层数过大导致栈空间不足，Expression::eval手动实现了求值栈。

所有的运算符都定义在通用的Expression类上，这样如果后续有更多无出度节点/终止节点加入，也可以通过向Expression转化轻松支持大量已有的运算符。

Node::getAllPreq和Node::getPreq分离，支持在性能上的优化。如果计算图的结构比较固定，可以通过调用getAllPreq获得所有依赖节点，预先建好图，利于静态优化。而getPreq可以分步返回依赖，使短路型的条件判断成为可能。

Node中含有version标记，对于计算密集型的节点，如果前驱节点没有更新过，可以不重复计算当前节点，提高效率。这个version标记的控制权完全在Node::eval手中，方便print等有特殊需求的节点自定义行为。

使用了std::shared\_ptr、std::map等STL，书写简洁且利于与现有库结合。

所有的浮点数类型都没有直接写成float，而是定义了别名num\_t。虽然后期不一定能更改，但是即使是调试时如果需要切换成别的类型来存储，也非常方便。