**代码互评**

**排序**

1. D组
2. E组
3. C组
4. B组

**评价**

**B组：**

优点：

1. 简单易懂

缺点：

1. 可扩展性不好，运算符未重载，后续实现功能时调用api较为困难；
2. 在“pc.h”中point poison[6000]确定了节点数，占用空间一定；
3. 图的逻辑与节点运算逻辑没有分离；
4. ifexist函数效率过低，可考虑用map一类的数据结构维护名称到编号的映射

**C组：**

优点：

1. 可读性较强
2. 节点类形成了较为progressive的结构，Node类下有单目、双目、三目三个子类，后续开发时可以根据需要选择自定义化的程度
3. 通过给节点增加时间戳，在重新计算的时候自动清空所有节点
4. 增加了三个额外功能，增添了计算图功能的丰富性

缺点：

1. 添加节点的时候需要经过Graph::InsertOperation这个函数，这个函数完成了所有判断。如果后续需要新加节点类型的话，必须要修改这个函数的实现，这不利于拓展。可选的改进方案为，将InsertOperation改成接收一个Node\*参数并将该节点加入图中，而解析字符串的代码独立出来到main.cpp中
2. Node节点的运算符重载返回的是临时计算结果的加减乘除结果。因为计算的last\_time不一定一样，结果也不一定有实际意义。将这样一个重载运算符加入API有些意义不明，容易造成后续开发人员的误解

**D组：**

优点：

1. 可读性较强
2. 数据节点类都是Data\_Node的子类，计算节点类是Operation\_Node的子类，而Data\_Node类、Operation\_Node又是Basic\_Node的派生类，继承结构清晰，可复用性高，可扩展性强

缺点：

1. Graph::initialize\_operator中有大量重复代码，可以适当抽象出来
2. 添加节点的操作与字符串解析写在了一起，耦合性过强，不适合在后续开发中在程序内调用接口；也不利于添加其他类型的节点

**E组：**

优点：

1. 支持多种数据类型的运算，方便后期开发时加入向量、矩阵的运算
2. 重定义了各类运算函数，调用方便

缺点：

宏定义过多，虽然简洁，但是可读性较差，难以维护，不符合oop追求可读性的原则，不利于后序人员理解及开发