中山大學 SUN YAT-SEN UNIVERSITY 程序设计(II)实验报告

学院: 数据科学与计算机学院　　专业：软件工程

姓名: 温卓沛　　学号： 17343115　　日期： 2018 年 07 月 07 日

实验题目：期末 Project—Simple Circuit

一、 代码运行方式

1. 编译方式：

直接将全部代码加入到 Microsoft Visual Studio 2017 一个新工程里，然后点击编译

2. 运行方式：

在 Microsoft Visual Studio 2017 编译后使用其运行按键

二、 操作方法

根据里面的操作指示，有：

1） 首先得输入你的用户名：

2） 然后有一些操作：

3） 添加：


F:\中大\黄方军老师\作业-大一下期末大作业-简单电路\Logic\Debug\Logic.exe
请选择要添加的逻辑门的类型：
1 - 与门
2 - 非门
3 - 或门
4 - 异或门
5 - 与非门
6 - 返回
请输入你的选择，并以ENTER键进行录入：

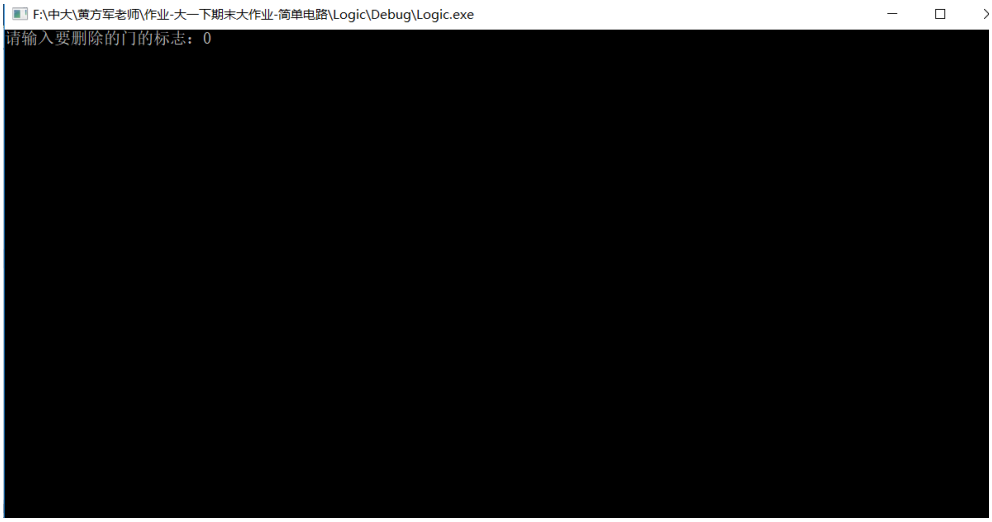
4） 连接：


F:\中大\黄方军老师\作业-大一下期末大作业-简单电路\Logic\Debug\Logic.exe
连接操作
请输入作为输入端的门的标志：0
请输入作为输出端的门的标志：0

5） 断开：


F:\中大\黄方军老师\作业-大一下期末大作业-简单电路\Logic\Debug\Logic.exe
断开操作
请输入作为输入端的门的标志：0
请输入作为输出端的门的标志：1

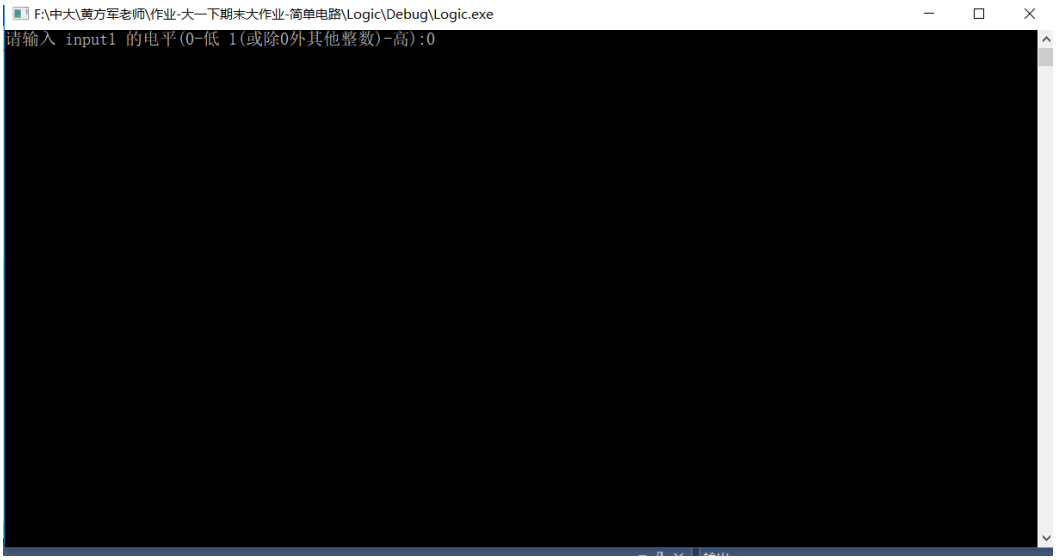
6） 删除：



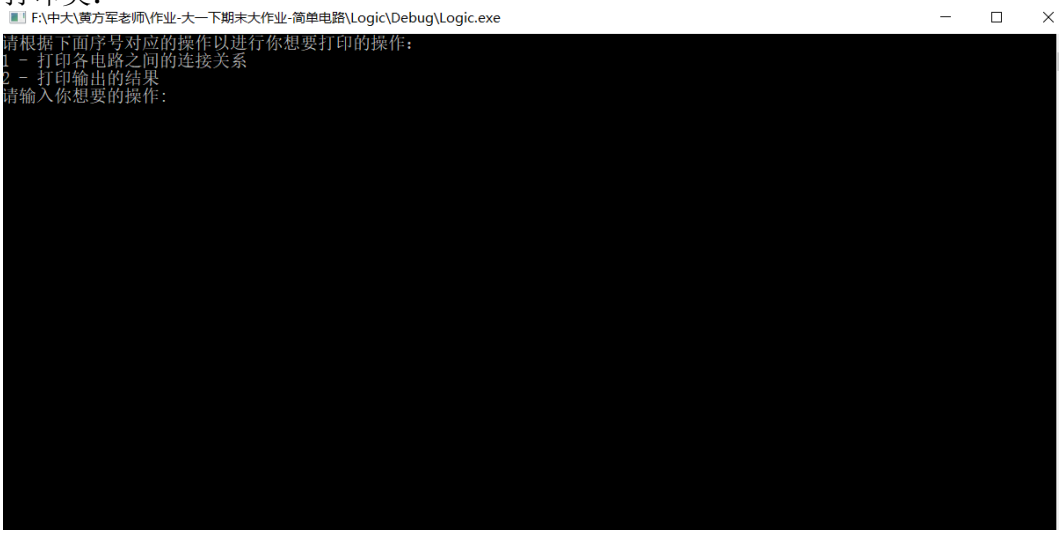
7） 设定输入和输出：

8） 检查环：

9) 运行(这个时候对输入进行录入)：



10) 打印类：



基本操作就如上了!!

三、 设计思路

1. 首先是对每个门进行**封装**(皆以一个抽象基类为父类进行**继承**，以实现**多态**)，封装实现改变内部数据成员的函数。
2. 然后使用一个 DOOR_LIST 类（不要介意是 DOOR 而不是 GATE，只因 GATE 看起来不太像是个类名..），对生成的门进行操作。
3. 最后使用一个 CLIENT 类封装一个 DOOR_LIST 类的实例，实现从 client 到实现操作的过程。
4. 上述 3 点是整体结构，而在门的连接中，我使用的是在各个门中使用一个 vector<DOOR*>的 STL 容器去进行对相连接的门进行连接。
5. 另外，使用继承于与各种门相同的基类的输入门与输出门类来进行对输出与输入的操控(在 DOOR_LIST 中添加装有输出门与输入门的 vector 容器进行标记)。

6. 对于搜索环或者进行运行操作，我使用的是递归版的 DFS。

7. 若有**环异常**，还可以将环异常中的门打印出；而对于无法正常运行的错误，则需要操作者自己检查连接去（所有的门必须要有相对于足够的输入—除了非门是只能等于 1 与 VCC、GUD、输入类、输出类没有输入外其他的门输入数至少为 2 个）。

8. 此外，考虑到进行输入操作以实现功能时采用的是数字 1.2.3……的操作，如果恶意输入不是数字的话，cin 可能会产生异常，故使用了异常处理使输入要一直输入到保证没错误的时候以达成**异常处理**，如图：

```
case 8:
{
    system("cls");
    print_3();
    int nnn;
    //输入异常处理
    int flags = 1;
    while (flags != 0) {
        std::cin >> nnn;
        try {
            if (std::cin.fail())
                throw 1;
            flags = 0;
        }
        catch (int) {
            std::cin.clear();
            std::cin.sync();
            std::cin.ignore();
            std::cout << "输入属于非数字，属于异常，请重新输入：";
        }
    }
    switch (nnn)
```

9. 至于**运算符重载**，我采用了对<<的重载，如图：

```
static int number_of_circuit;
friend std::ostream &operator<<(std::ostream &output, const CLIENT &D) {
    output<<  "Goodbye , " << D.name<<" .Good luck to you!!\n" ;
    return output;
}
```

```
case 9:
{
    this->door_list->remove_all();
    std::cout << *this << std::endl;
    flag = 1;
}
    break;
```

10. **一位全加器的代码连接方式在最下面的附录！**

四、 拓展

1. 能够实现在门连接后的断开与删除（要求上只要求连接）。
2. 实现更多的门，如 NAND（该门可实现所有其他门的功能，也可以用它来进行封装其他门）。

五、 待改进的不足

1. 代码总长较长，各种门的多态函数功能重复，应置于基类，然我并没有。
2. 使用了较多的 STL 容器，占用内存空间较多，且使用深搜较耗时。

六、 收获

完成本次 project 使我对类继承与多态的使用有更深的理解与认识，也对异常处理的妙用有了更多的感触；同时也了解了一些不同的算法，如拓扑排序、DFS、BFS、并查集等，为以后的学习定下基调。

七、 全加器代码实现

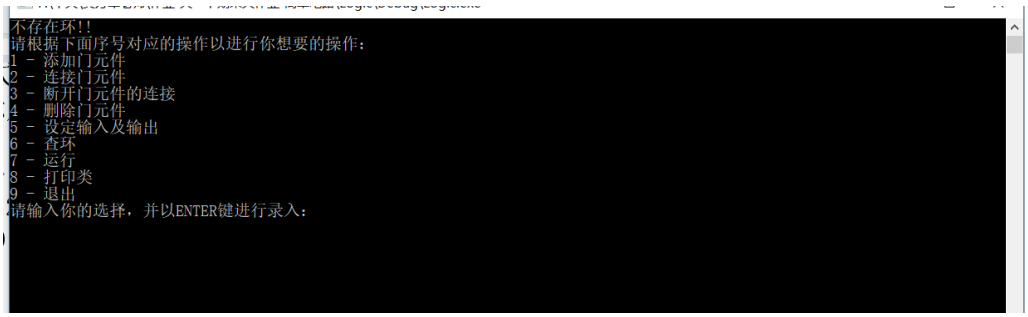
从输入昵称后开始：

连接实现：

```
1 1 1 1 1 4 1 4 1 3 5 1 in1 5 1 in2 5 1 cin 5 2 sum 5 2
cout 2 -3 2 2 -4 2 2 2 3 2 -5 3 2 2 0 2 -5 0 2 -3 1 2 -4 1
2 3 -6 2 0 4 2 1 4 2 4 -7
```

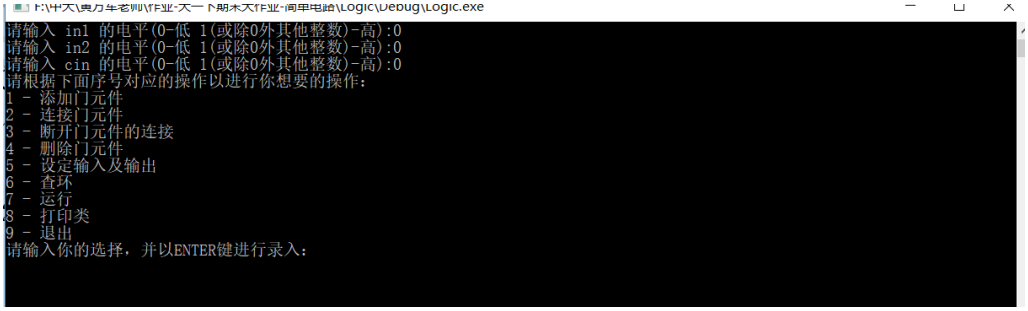
查环实现：

6



运行实现：

7 in1(这一位的数 1)的电平 in2(这一位的数 2)的电平 cin(上一位的进位)的电平



查看结果实现：
8 2

```
请根据下面序号对应的操作以进行你想要打印的操作：
1 - 打印各电路之间的连接关系
2 - 打印输出的结果
请输入你想要的操作:2
输入类:
in1 : 0
in2 : 0
cin : 0
输出类:
sum : 0
cout : 0
请根据下面序号对应的操作以进行你想要的操作：
1 - 添加门元件
2 - 连接门元件
3 - 断开门元件的连接
4 - 删除门元件
5 - 设定输入及输出
6 - 查环
7 - 运行
8 - 打印类
9 - 退出
请输入你的选择，并以ENTER键进行录入：
```