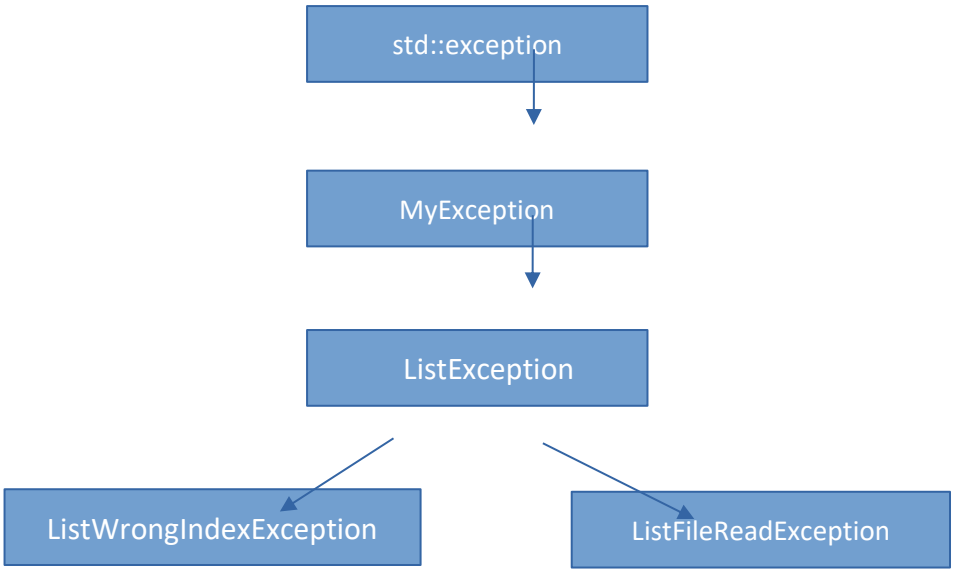
Наследование. Множественное наследование. Обработка исключений.

Для данного задания использовать классы из 1-ой лабораторной согласно своему варианту:

1. **Базовое наследования.**
   Класс списка (стек, очередь) разделить на два класса:
   - базовый класс неупорядоченный однонаправленный или двунаправленный список (по заданию 1-ой лаб.);
   - дочерний класс стек, очередь или упорядоченный список (по заданию).
   Например, в задании «двусвязный стек», надо реализовать базовый класс ДвусвязныйСписок и унаследовать от него класс ДвусвязныйСтек.
   Т.е. существующий класс разделить на базовый и наследник.
   В дочернем классе скрыть ненужные методы, определить новые названия функций.
   Например, в базовом классе есть функции типа add, addEnd, addBegin, а в стеке - pop, push
2. Добавить поддержку отрицательных индексов ([-1] – это последний элемент, [-2] – предпоследний и т. д.) для операции индексации в базовом классе.
3. **Множественного наследование на классе данных.**
   Необходимо создать два базовых класса:
   - класс Data (это ваш класс с данными: студент, книга и т. п.)
   - класс Node, содержащий в качестве свойств поля типа next, prev и необходимые виртуальные функции, используемые в классе List. Класс Node должен быть абстрактным, т. е. содержать минимум одну чисто виртуальную функцию..

   Создать дочерний класс NodeData (наследует от классов Node и Data). Продемонстрировать использование виртуальных функций.

   Класс списка (List — ваш базовый класс из 1-го пункта) работает только с типом Node. Во входных и возвращаемых типах, внутри функции могут быть только объекты класса Node. Фактически необходимо передавать в методы объекты типа NodeData, благодаря механизму виртуальных функции можно обращаться к методам базового класса.
4. **Обработка исключений.**
   Добавить иерархию классов-исключений для возможных типов ошибок в ваших классах списков и данных. Во всех местах, где могут быть ошибки, возбуждать исключения согласно логике. Перехват и обработку исключений сделать в функции main.
   Собственный базовый класс исключений унаследовать от стандартного класса std::exception и перегрузить в нем функцию what.
   Пример иерархии:
   MyException: public exception
   ListException: public MyException
   ListWrongIndexException: public ListException
   ListFileReadException: public ListException


std::exception
MyException
ListException
ListWrongIndexException
ListFileReadException