1. 强化训练-数组类封装
   1. 设计类 myArray
   2. 属性
      1. int m\_Capacity数组容量
      2. int m\_Size 数组大小
      3. int pAddress 维护真实在堆区创建的数组指针
   3. 行为
      1. 默认构造
      2. 有参构造
      3. 拷贝构造
      4. 析构
      5. 根据位置 设置数据
      6. 根据位置 获取数据
      7. 尾插
      8. 获取数组容量
      9. 获取数组大小
2. 加号运算符重载
   1. 对于内置的数据类型，编译器知道如何进行运算
   2. 但是对于自定义数据类型，编译器不知道如何运算
   3. 利用运算符重载 可以让符号有新的含义
   4. 利用加号重载 实现p1 + p2 Person数据类型相加操作
   5. 利用成员函数 和 全局函数 都可以实现重载
   6. 关键字 operator +
   7. 成员本质 p1.operator+(p2)
   8. 全局本质 operator+(p1,p2)
   9. 简化 p1 + p2
   10. 运算符重载 也可以发生函数重载
3. 左移运算符重载
   1. 不要滥用运算符重载，除非有需求
   2. 不能对内置数据类型进行重载
   3. 对于自定义数据类型，不可以直接用 cout << 输出
   4. 需要重载 左移运算符
   5. 如果利用成员 函数重载 ，无法实现让cout 在左侧，因此不用成员重载
   6. 利用全局函数 实现左移运算符重载
      1. *ostream*& operator<<(*ostream* &cout, Person & p1)
   7. 如果想访问类中私有内存，可以配置友元实现
4. 递增运算符重载
   1. 前置递增
      1. MyInter& operator++()
   2. 后置递增
      1. MyInter operator++(**int**)
   3. 前置++ 效率高于 后置++ 效率 ，因为后置++会调用拷贝构造，创建新的数据
5. 指针运算符重载
   1. 智能指针
   2. 用途： 托管new出来的对象的释放
   3. 设计smartPoint智能指针类，内部维护 Person \* ，在析构时候释放堆区new出来的person对象
   4. 重载 -> \* 让 sp智能指针用起来向真正的指针
6. 赋值运算符重载
   1. 编译器会默认个一个类添加4个函数
      1. 默认构造、析构 、 拷贝构造（值拷贝） 、 operator=（值拷贝）
   2. 如果类中有属性创建在堆区，利用编译器提供的 = 赋值运算就会出现 堆区内存重复释放的问题
   3. 解决方案：利用深拷贝 重载 =运算符
   4. Person& operator=( const Person &p)
7. []运算符重载
   1. int& operator[](int index);
   2. 实现访问数组时候利用[] 访问元素