1. 数组类封装
   1. 提供int类型数组
   2. 属性
      1. int \*pAddress 堆区数组指针
      2. int m\_Capacity 数组容量
      3. int m\_Size 数组大小
   3. 行为
      1. 默认构造
      2. 有参构造
      3. 拷贝构造
      4. 尾插
      5. 根据位置设置数据
      6. 根据位置获取数据
      7. 获取数组容量
      8. 获取数组大小
      9. 析构
2. 加号运算符重载
   1. 对于内置数据类型 ，编译器知道该如果进行运算
   2. 但是对于自定义数据类型，编译器并不知道该如果运算
   3. 利用运算符重载 可以解决问题
   4. 实现两个Person数据类型相加
   5. 分别利用 成员函数 和 全局函数 来做实现
      1. 成员函数 本质 p1.operator+( p2)
      2. 全局函数 本质 operator+(p1,p2)
      3. 简化为 p1 + p2
   6. 运算符重载 也可以发生 函数重载
3. 左移运算符重载
   1. 对于内置数据类型 ，编译器知道如果 cont进行<<运算符输出
   2. 对于自定义数据类型，无法输出
   3. 重载左移
      1. 利用成员函数： 失败，原因不能让cout在左侧
      2. 利用全局函数： ostream & operator<< ( ostream & cout , Person &p )
      3. 如果Person 属性是私有，可以配合友元技术访问私有属性
4. 递增运算符重载
   1. 前置++
      1. 返回引用 MyInt & operator++()
   2. 后置++
      1. 返回值 MyInt operator++(int)
   3. 前置++的效率会略高于 后置++
      1. 原因是前置不会调用拷贝构造
5. 智能指针案例 指针运算符重载
   1. 智能用途 用来托管堆区创建的对象的释放
   2. 如果想让sp对象当做一个 指针去对待，需要重载 -> \*
6. 赋值运算符重载 =
   1. 系统会默认给一个类添加4个函数 ： 默认构造、拷贝构造、析构、operator=
   2. 由于系统提供的operator= 会进行简单的值拷贝 ，导致如果属性中有堆区的数据，会进行重复释放
   3. 解决方案：需要重载 operator =
   4. 返回值 Person& operator=( const Person & p)
7. []运算符重载
   1. 上午数组类封装中将对象 当数组名对待，可以利用[]方式对数组中的数据进行设置和获取
   2. int& operator[](int index);