1. 加号运算符重载
   1. 如果想让自定义数据类型 进行+运算，那么就需要重载 + 运算符
   2. 在成员函数 或者 全局函数里 重写一个+运算符的函数
   3. 函数名 operator+ () {}
   4. 运算符重载 也可以提供多个版本
2. 左移运算符重载
   1. 不要随意乱用符号重载
   2. 内置数据类型 的运算符不可以重载
   3. cout << 直接对Person自定义数据类型 进行输出
   4. 写到全局函数中 ostream& operator<< ( ostream & cout, Person & p1 ) {}
   5. 如果重载时候想访问 p1的私有成员，那么全局函数要做Person的友元函数
3. 前置 后置 ++ 运算符重载
   1. 自己实现int类型 MyInteger
   2. 内部维护以int数据
   3. MyInteger myInt
   4. myInt ++ 后置 ++myInt 前置
   5. 重载++运算符 operator++() 前置 operator++(int) 后置
   6. 前置理念 先++ 后返回自身 后置理念 先保存住原有值 内部++ 返回临时数据
   7. 练习 自己实现递减运算符重载 --
4. 智能指针实现
   1. Person类有showAge 成员函数
   2. 如果new出来的Person对象，就要让程序员自觉的去释放 delete
   3. 有了智能指针，让智能指针托管这个Person对象，对象的释放就不用操心了，让智能指针管理
   4. 为了让智能指针想普通的Person\*指针一样使用 就要重载 -> 和\*
5. 赋值运算符重载
   1. 系统默认给类提供 赋值运算符写法 是简单值拷贝
   2. 导致如果类中有指向堆区的指针，就可能出现深浅拷贝的问题
   3. 所以要重载 = 运算符
   4. 如果想链式编程 return\*this
6. []运算符重载
   1. 返回数组索引的引用
   2. int & operator[](int index)
   3. return this->pAddress[index]