作业七

概念题

1. 什么是操作符重载？操作符重载有哪些基本原则？

```
操作符重载允许为自定义类型（通常是类或结构体）定义或修改语言预定义操作符（如加号 +、乘号 * 等）的行为。在 C++ 中，操作符重载是一种常用的技术，它提供了一种灵活而表达力强的方式来定义类的行为。
遵守对称性：对于对称的操作符（如 +、-、*、/），如果类设计是对称的，那么操作符也应该对称地重载。
友元声明：可以声明操作符函数为类的友元，以便它能够访问类的私有成员。
不重载以下操作符：一些操作符如成员访问 .、指针访问 -> 和条件操作符 ?:
```

2. 简述操作符重载的两种实现途径，这两种实现途径有哪些异同？

```
成员函数实现
操作符重载可以作为类的成员函数实现。这是最常见的实现方式，特别是对于那些操作与类本身的一个对象紧密相关的情况。

非成员函数实现（友元函数）
操作符重载也可以通过非成员函数实现，特别是当操作符需要以对称的方式或以不直接涉及类成员变量的方式操作时。在这种情况下，操作符函数可以是类的友元函数。

参数顺序：

成员函数：隐式地使用 this 指针表示第一个操作数。
非成员函数：需要显式地传递所有操作数。

友元关系：

成员函数：不需要友元声明。
非成员函数：必须被类声明为友元，以便访问类的非公共成员。

对称性：

成员函数：通常不能直接处理不涉及类对象的操作（如 int + MyClass）。
非成员函数：可以自然地处理涉及类类型和内置类型的操作，如 int + MyClass。
```

3. 请从实现和使用的两个方面，比较前置和后置的 operator++ 的不同

```
前置形式 (operator++):

返回类型通常是类的引用（*this），因为前置形式需要在递增后返回对象本身。
实现时通常会使用一个额外的成员变量来保持递增前的状态，或者通过其他方式记住递增前的位置。

后置形式 (operator++(int)):

返回类型通常是通过值返回的，并且通常返回一个迭代器类型的副本。
后置形式接受一个额外的整数参数（通常是一个特殊的值，如 0），这个参数没有实际用途，仅是为了在语法上与前置形式区分。
```

编程题

以下编程题为课堂实验，请在教学平台-课堂实验中完成

任务描述

迭代器是容器类（如数组，链表，队列，栈等）提供的、用于遍历和访问容器内对象的辅助类。在本题中，我们考虑一种简化的迭代器，它为我们自定义的整型数组类提供元素的遍历和访问。
自定义的整型数组类如下：

```
class MyArray{
    int * arr;
    int size;
public:
    MyArray(int size);
    ~MyArray();
};
```

它根据参数创建和维护特定长度的动态整型数组，并且提供迭代器`Iterator`来访问数组`arr`中的元素。由于迭代器常与特定类关联，所以我们使用`MyArray`的内部类来实现`Iterator`，定义如下：

```
class MyArray {
    int * arr;
    int size;
public:
    class Iterator{
        ...
    public:
        ...
        bool get(int &value) const;
        bool put(int value);
        ...
    };
public:
    MyArray(int size);
    ~MyArray();
    Iterator begin();
    Iterator end();
}
```

可以看到，`MyArray`类新增了两个接口`Iterator begin()`和`Iterator end()`来提供两个特殊的迭代器对象：

- 每个迭代器对象都提供两个接口`bool get(int &value)`和`bool put(int value)`
  - `bool get(int &value)`：获取当前迭代器所指向的元素
    - 如果迭代器位于一个合法的位置：`arr[0]`-`arr[size-1]`，则通过`value`返回对应元素，返回值为`true`
    - 否则不改变`value`的值，返回`false`
  - `bool put(int value)`：向迭代器指向的元素存放值`value`
    - 如果迭代器位于一个合法的位置：`arr[0]`-`arr[size-1]`，则将对应数组元素的值更新为`value`，返回`true`

        - 否则不改变任何值，返回`false`
- `Iterator begin()`：返回的迭代器对象，指向数组的第一个元素`arr[0]`
    - 具体而言，下列代码中`iter.get(v)`获取的应该是`ma.arr[0]`的值

```
MyArray ma(10);
MyArray::Iterator iter = ma.begin();
iter.put(20); // true, set ma.arr[0]=20
int v = -1;
iter.get(v); // true, v=20
```

    - 当然，你需要自己思考，如果`MyArray`类中的`arr`数组为空（长度为0），`Iterator begin()`返回的迭代器应该指向哪儿？
- `Iterator end()`：返回的迭代器对象，指向数组`arr`的末尾，但不是任何数组中的元素
    - 末尾实际上仅仅起到一个标记遍历结束的作用，它并不对应任何数组元素。当然你可以把它设置成指向`arr[size]`，但注意用该迭代器访问数组元素(`get`和`put`)是无法成功且没有意义的。
    - 具体而言，我们通常会看到迭代器的如下用法，实现遍历容器中的每个元素：

```
for(MyArray::Iterator iter = ma.begin(); iter != ma.end(); iter++){
    int v;
    iter.get(v);
    ...
}
```

      通过判断迭代器`iter`是不是迭代到了容器的末尾，让整个遍历可以结束。

或许你已经意识到，为了使迭代器更加方便使用，我们需要为其重载一些操作符，它们的含义如下：

- `operator++`
    - 将迭代器指向数组中下一个相邻元素
    - 考虑前置和后置
- `operator--`
    - 将迭代器指向数组中上一个相邻元素
    - 考虑前置和后置
- `operator==`
    - 判断两个迭代器是否指向同一个位置（元素）
- `operator!=`
    - 与上一个操作符相反的语义
- `operator+(int len)`
    - 将迭代器移动到当前元素向后的第`len`个元素
- `operator-(int len)`
    - 将迭代器移动到当前元素向前的第`len`个元素

注意

- 数组的第一个元素`arr[0]`和最后一个元素`arr[size-1]`并不相邻（数组不是环状的），所以你需要考虑上述操作符在跨过数组前后边界时应该如何处理。要求只有一个：跨过数组边界的迭代器指向非法位置，这些迭代器不能用于访问数组元素——调用`get`和`put`时返回值为`false`。
- 元素`arr[size-1]`的下一个"相邻元素"是末尾（也就是`end()`迭代器指向的元素），末尾的上一个相邻元素是`arr[size-1]`。这样的设计会使通过迭代器从头到尾遍历元素更自然。

你的任务

- 完善`MyArray`类中的4个成员函数。**注意：`MyArray`类的对象析构时，其所有的迭代器也应随之失效——调用`get`和`put`时返回值为`false`。**
- 自定义`Iterator`类的数据成员，并添加相应的构造和析构函数。实现`get`和`put`两个成员函数，并添加前述所列的操作符重载，使得上述关于`Iterator`的功能可以满足。

（仅作）思考

你或许已经发现，通过迭代器访问容器中的对象和通过指针访问非常的相似，能否通过操作符重载进一步简化`get`和`put`？通过迭代器访问相比于通过指针访问，有哪些优劣？

测试说明

- 测试框架仅会测试上述要求实现的功能
- 测试框架虽然是针对单个功能设置的测试点，但若你未提供所有需要实现的函数/操作符的定义，将无法通过编译，导致测试失败。所以尽管你尚未明白某个函数/操作符的具体实现，也请定义一个伪实现，以确保其它部分功能正常测评
- 测试点说明
  - 每个测试点都会依赖begin，end，get，put
  - case1-2：测试基本的get、put
  - case3：测试MyArray析构之后的get，put
  - case4：MyArray中动态数组为空（长度为0）
  - case5：operator+，注意操作符语义，返回值类型，不要修改自己
  - case6：operator-
  - case7：operator++，注意前置和后置的语义
  - case8：operator--
  - case9：operator!=
  - case10: operator==
  - case11: 综合测试

提交注意事项

截止时间：2023-4-23 23:59

文件格式：姓名-学号.pdf

提交方式：南大计科在线实验教学平台

请同学们于截止时间前在南大计科在线实验教学平台上提交，每次作业最终只需要提交一个pdf文件即可，以“姓名-学号.pdf”的方式命名。

注意：

- 请按要求命名文件，并且只提交一个PDF文件，~~编程题代码请附在PDF中。~~ 任何错误的命名和文件格式将影响你的作业得分。
- 本次作业编程题不需要在pdf中提交，请在教学平台-课堂实验下完成。