```
double & DoubleArray::operator [](int index)
{
    if(index<min_index || index>max_index)  //若下标越界显示出错信息并退出
    {
        cout<<"Index out of range";
        exit(1);
    }
    return data[index-min_index];
}

int main()
{
    DoubleArray x(3,8);
    for(int i=3; i<=8; i++) x[i]=i*i;
    x[4]=-x[4];
    for(int j=3; j<=8; j++) cout<<x[j]<<' ';
    cout<<endl;
    return 0;
}
```

运行结果：

```
9 -16 25 36 49 64
```

思考与练习：

模仿本例，设计一维 int 数组类 IntArray，创建其对象时需要指定数组元素的个数，数组的下标从 1 开始。

小　结

本章介绍了使用运算符函数实现自定义类型运算的方法，详细介绍了运算符重载的概念和两种具体的实现方法：重载为类的成员函数和重载为友元函数，并结合具体的实例说明了常见一元运算符和二元运算符的重载方法和应注意的问题。建议读者在认真研究本章每一个例题的基础上，根据“思考与练习”的建议，举一反三，进行拓展练习，掌握 C++中运算符重载的两种基本方法。

习　题

一、选择题

1. 如果表达式--j*k 中的“--”和“*”都是作为友元重载的运算符，采用运算符函数调用格式，该表达式还可表示为（　　）。

A. operator*(j.operator--() , k)　　B. operator*(operator--(j) , k)

C. j.operator--().operator*(k)　　D. k.operator*(operator--(j))

2. 在下列运算符中，只能作为二元运算符重载的是（　　）。

A. +　　B. -　　C. *　　D. /

3. 假定为类 X 成功地重载了++、=、-和[]等运算符，则其中肯定属于成员函数的运算符是（　　）。

A. -和=　　B. []和++　　C. =和[]　　D. -和[]

4. 在重载一个运算符时，其运算符函数的参数表中没有任何参数，这说明该运算符函数是（　　）。

A. 将一个一元运算符重载为全局函数
B. 将一个一元运算符重载为成员函数
C. 将一个二元运算符重载为全局函数
D. 将一个二元运算符重载为成员函数

5. 有以下程序：

```
#include <iostream>
using namespace std;

class MyString {
    char ss[80];
public:
   MyString(const char* s) { strcpy(ss, s); }
   MyString& operator += (const MyString &a)
 {
      strcat(ss, a.ss);
      return *this;
 }
      const char *getString()const{ return ss; }
 };

 int main()
 {
     MyString x("abc"), y("cde");
     cout << (x+=y).getString() << endl;
     return 0;
 }
```

执行后的输出结果是（　　）。

A. abcabc　　B. abccde　　C. "abcabc"　　D. "abccde"

6. 已知针对类 AX 定义了一个唯一的运算符函数，使得对于 AX 的对象 ax，表达式 ax-5 和 5-ax 都能正确求值。有鉴于此，下列表述中错误的是（　　）。

A. 该运算符函数是类 AX 的成员函数
B. 该运算符函数的形参表中有两个参数
C. 该运算符函数的函数名是 operator-
D. 利用类 AX 的构造函数可将 5 转换成一个 AX 对象

7. 从下列运算符函数的原型看，肯定不属于类 FunType 的成员函数的是（　　）。

A. int operator + (FunType);　　B. FunType operator + ();

C. FunType operator + (int);　　D. int operator + (FunType, FunType);

8. 关于运算符重载，下列表述正确的是（　　）。

A. 可以针对基本类型的数据，重新定义“*”的含义

B. 通过运算符重载可以改变运算符的操作数个数

C. 只能重载 C++中已有的运算符

D. C++中所有的运算符都可以重载

9. 下列运算符重载函数的原型中，错误的是（　　）。

A. FunType operator * (FunType, FunType);

B. FunType operator - (FunType);

C. FunType& operator += (FunType&, FunType);

D. FunType& operator = (FunType&, FunType);

10. 有如下函数原型：

```
FunType operator*(FunType)const;
```

针对这个函数原型，下列表述错误的是（　　）。

A. 这是运算符 * 的重载函数

B. 这是一个成员函数

C. 被重载的运算符是一个一元运算符

D. 这个函数不改变类的任何数据成员的值

11. 在为类 FunType 重载运算符 += 时，重载函数的最后一个语句是

```
return *this;
```

关于这个重载函数，下列表述中错误的是（　　）。

A. 它的名称是 operator+=

B. 它的返回值类型是 FunType 或 FunType&

C. 它的形参表只包含一个形参定义

D. 它是类 FunType 的友元函数

12. 在下面的重载运算符函数的原型中，错误的是（　　）。

A. FunType operator - (FunType, FunType);

B. FunType operator - (long, long);

C. FunType operator - (FunType, long);

D. FunType operator - (long, FunType);

二、填空题

1. 重载的运算符仍然保持其原来的优先级、操作数个数和_____不变。

2. 如果表达式 a=b*c 中的“*”是作为成员函数重载的运算符，采用函数名 operator*，该表达式还可以表示为_____。

3. 如果表达式 a=b-c 中的“-”是作为非成员函数重载的运算符，采用函数名 operator-，该表达式还可以表示为_____。

4. 如果表达式 k--中的“--”是重载的成员运算符，采用函数名 operator--，该表达式还可

以表示为_____。

5. 若为类 Bounce 重载友元运算符<,其操作结果为一逻辑型数据，则该运算符函数的原型是_____。

6. 有些运算符既可作为成员函数重载，也可以作为全局函数重载。在一个重载的运算符函数的定义中成功地使用了 this 指针，说明它是作为_____重载的。

7. 下面的函数定义试图以最符合运算符原有特性的方式重载 +=，请将下划线处缺失部分补充完整。

```
_____(Box& box1,const Box& box2)
{
  box1.balls+=box2.balls;
  return _____;
}
```

8. Person 对象的大小由它的 int 型数据成员 age 的大小确定，age 越大，相应的 Person 对象就越大。下面的函数重载了运算符 > ，它比较两个 Person 对象的大小，当第 1 个对象大于第 2 个对象时返回 true，否则返回 false。请将下划线处缺失部分补充完整。

```
_____ Person::operator>(Person p)
{
   return _____ ;
}
```

9. 针对 Person 对象的后置--操作就是将其 int 型数据成员 age 的数值减 1，返回一个与原对象（所做操作之前的对象）相同的 Person 对象。下面的函数作为 Person 的友元重载了运算符后置--，请将下划线处缺失部分补充完整。

```
Person operator--(_____)
{
  Person p0=p;
  p.age--;
  return _____;
}
```

10. 下面是作为 Person 的成员重载的类型转换运算符 const char *，它把 Person 对象转换成一个字符串，其中包含了数据成员 name(姓名，一个 C 字符串)和 age(年龄，一个 int 型数据)的信息。请将下划线处缺失部分补充完整。

```
operator _____()const{
    static char buf[80],s[20];
    strcpy(buf,"姓名: ");
    strcat(buf,name);
    strcat(buf,", 年龄: ");
    strcat(buf,itoa(age,s,10));
    return _____;
}
```

11. 下面的函数定义试图以最符合运算符原有特性的方式重载 +=，请将下划线处缺失部分补充完整。

```
Box& Box::operator+=(_____)
{
```

```
    balls+=box.balls;
    return _____;
}
```

12. 两个 Date 对象相等是指它们的 3 个 int 型成员 year、month 和 day（分别表示年、月、日）分别相等。下面的函数重载了运算符 ==，它判断两个 Date 对象是否相等，若相等则返回 true，否则返回 false。请将下划线处缺失部分补充完整。

```
_____ operator==(Date d1,Date d2)
{
  return _____ ;
}
```