一维数组基本操作的 C++代码

一维数组类

```
// 数组的初始构造函数
template<class T>
Array<T>::Array(int sz)
{
	if(sz<=0)
	{
		cout<<"Invalid Array Size."<<endl;
		exit(1);
	}
	size=sz;
	alist=new T [sz];
};

// 数组的复制构造函数
template<class T>
Array<T>::Array(const Array<T>&v)
{
	size=v.size;
	alist=new T [size];                          // 为复制数组申请空间
	for(int i=0; i<size; i++)                    // 复制数组元素
		alist[i]=v.alist[i];
};

// 重载下标符运算符 []
template <class T>
T & Array<T>::operator [] (int i) const
{
	if(i<0||i>=size)
	{
		cout<<"invalid index."<<endl;
		exit(1);
	}
	return alist[i];
};

// 重载赋值运算符＝
template <class T>
Array<T>& Array<T>:: operator = (const Array<T>&v)
{
	// 确定是否自我赋值
	if(this!=v)
	{
		size=v.size;
		delete [] alist;                 // 释放原空间
		alist=new T [size];          // 申请新空间
		for(int i=0; i<size; i++)  // 复制数组元素
			alist[i]=v.alist[i];
	}
	return *this;
};
```

```
// 重载一元减法运算符
template<class T>
Array<T> Array <T>:: operator - ( ) const
{
	Array<T>w(size);
	for(i=0; i<size; i++)
		w.alist[i]=-alist[i];
	return w;
}

// 重载二元减法运算符－
template<class T>
Array<T> Array<T>:: operator - (const Array<T>&v) const
{
	if(size!=v.size)
	{
		cout<<"invalid index."<<endl;
		exit(1);
	}
	Array<T>w(size);
	for(int i=0; i<size; i++)
		w.alist[i]=alist[i]-v.alist[i];
	return w;
};
```