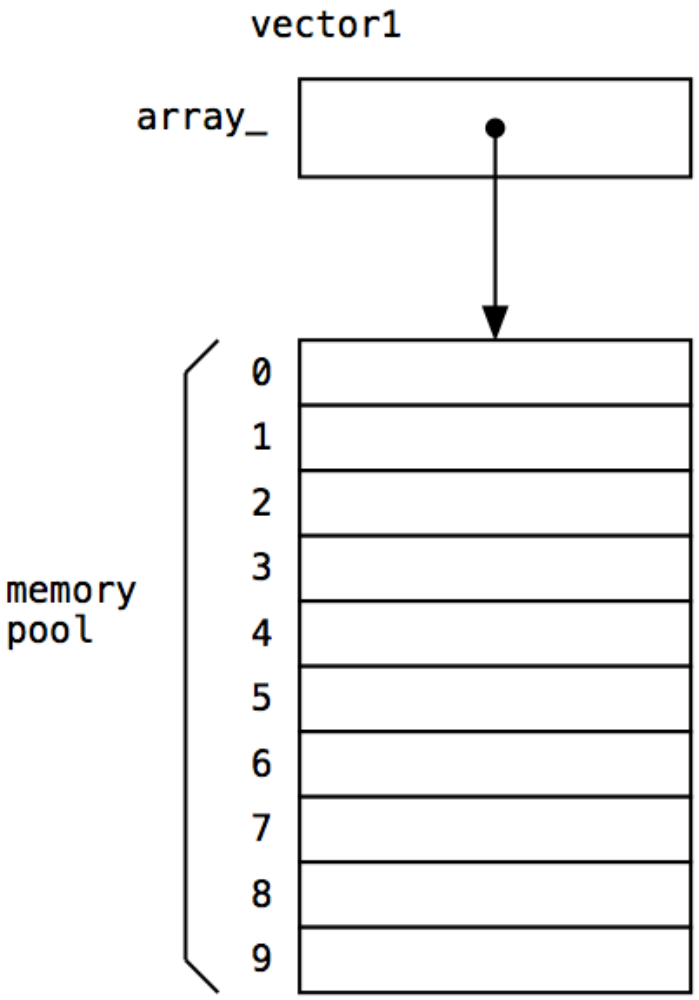
C++ Classes

```
class FixedVector
{
    public:
        FixedVector();
        FixedVector(int size);
        FixedVector(const FixedVector& a);
        ~FixedVector();
        int   Get(int index);
        voidSet(int index, int value);
        int   Size();
        FixedVector& operator=(const FixedVector& a);
    private:
        int   size_;
        int* array_;
};
```

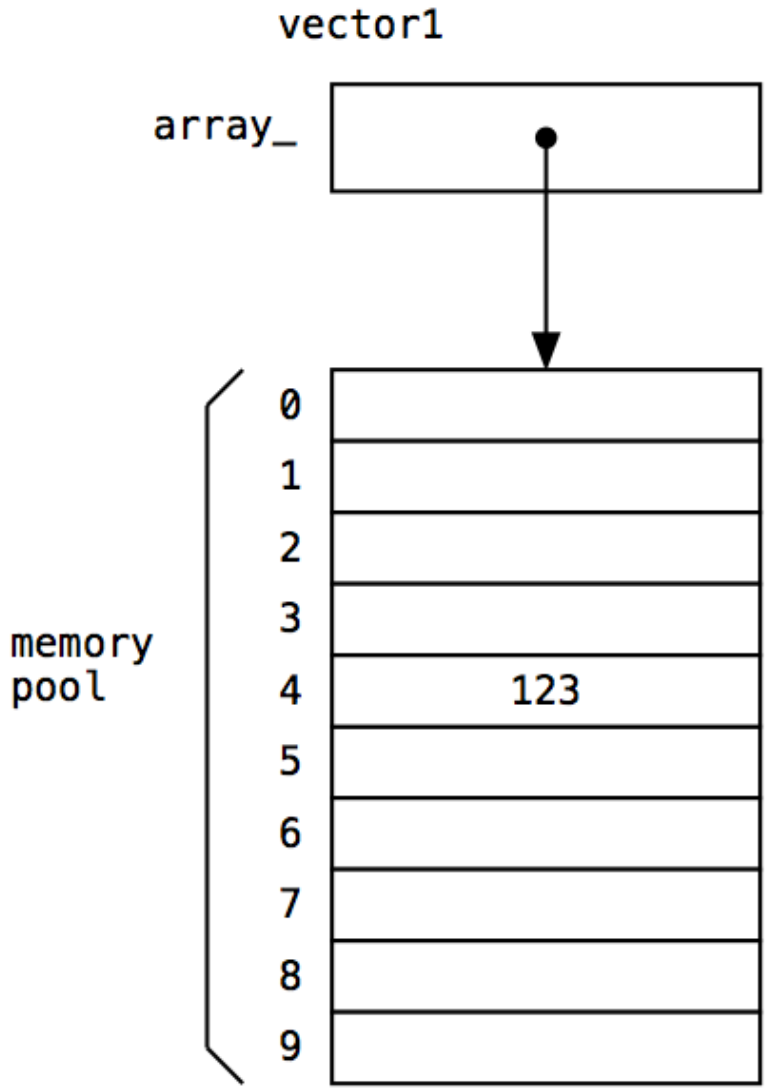
```
#include "FixedVector.h"
#define kDefaultSize (10)
FixedVector::FixedVector()
{
	size_ = kDefaultSize;
	array_ = new int[size_];
}
FixedVector::FixedVector(int size)
{
	size_ = size;
	array_ = new int[size_];
}
FixedVector::FixedVector(const FixedVector& a)
{
	size_ = a.size_;
	array_ = a.array;
}
```

```
FixedVector::~FixedVector()
{
	delete[] array_;
}
int FixedVector::Get(int index)
{
	return(array_[index]);
}
void FixedVector::Set(int index, int value)
{
	array_[index] = value;
}
```

FixedVector vector1;

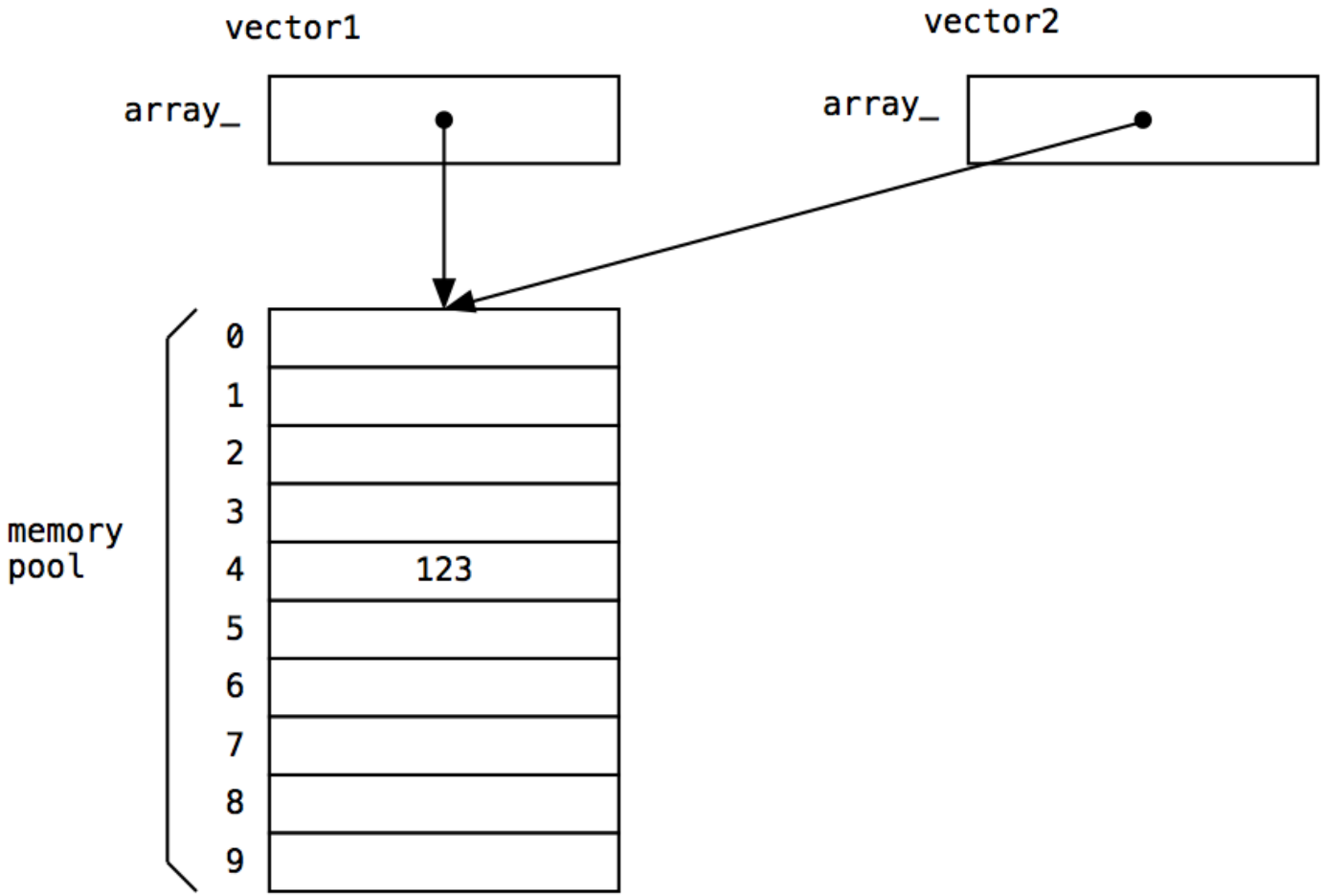


vector1.Set(4, 123);

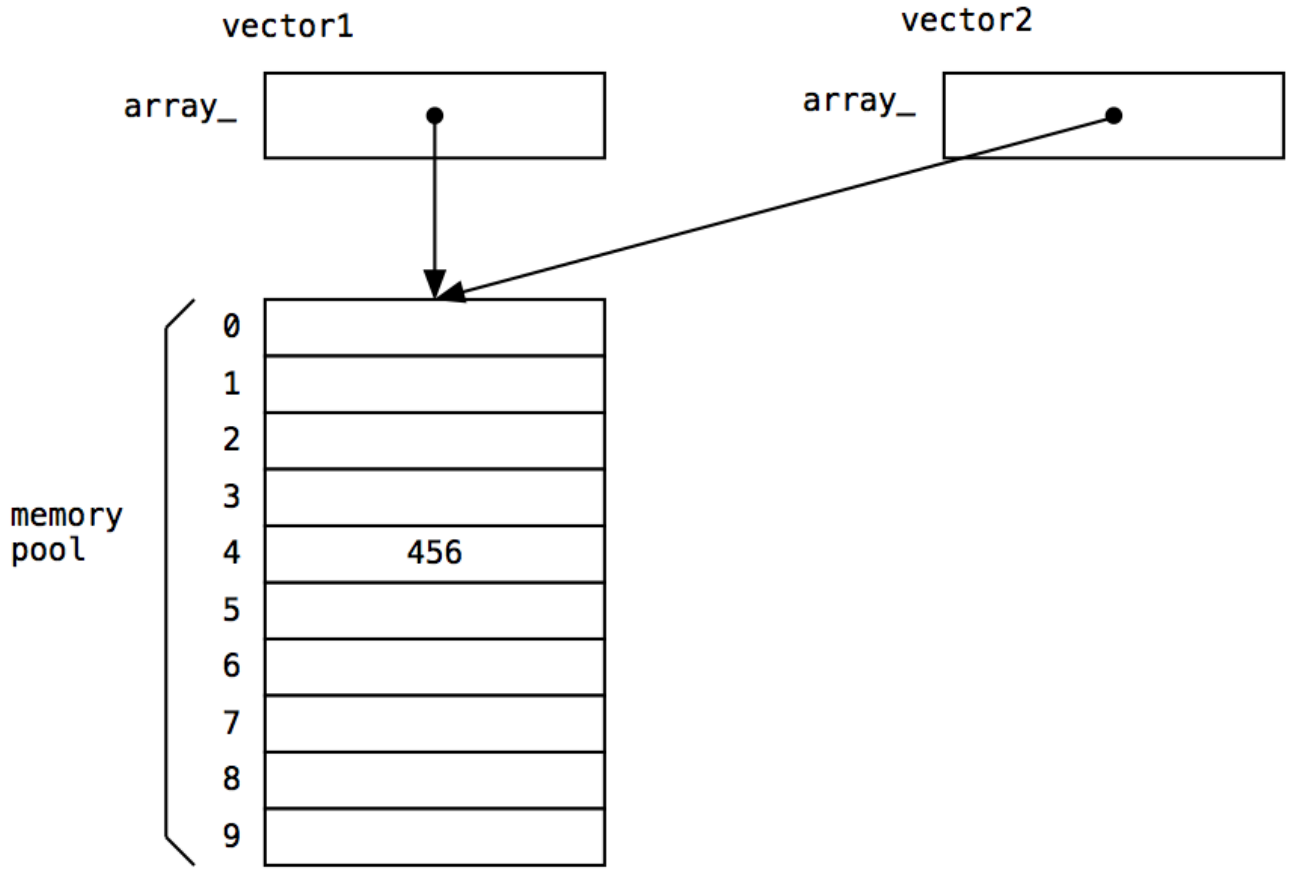


FixedVector vector2;

vector2 = vector1;



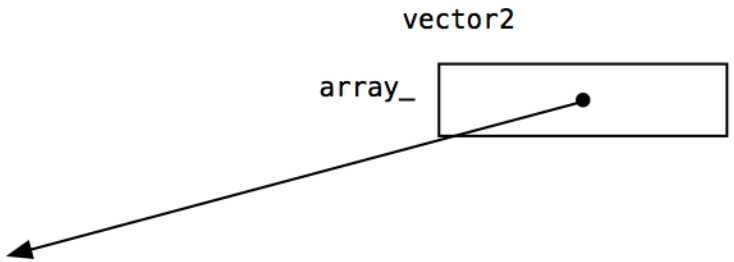
vector2.Set(4, 456);



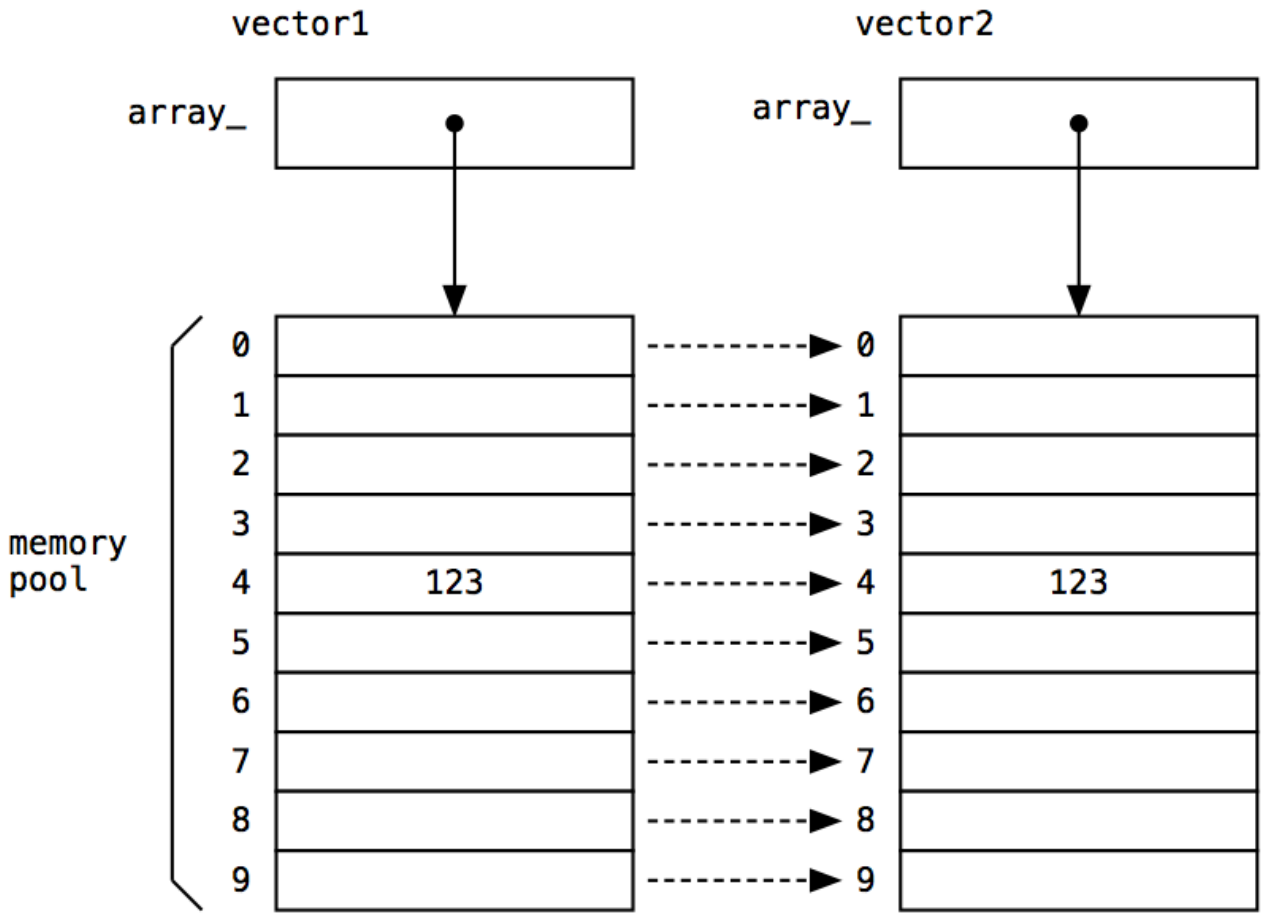
- Function returns to caller
- vector1 goes out of scope
- vector1's destructor is called
- delete [] array_ // in vector1
- vector2 goes out of scope
- vector2's destructor is called
- delete [] array_ // in vector2

but array is gone because of

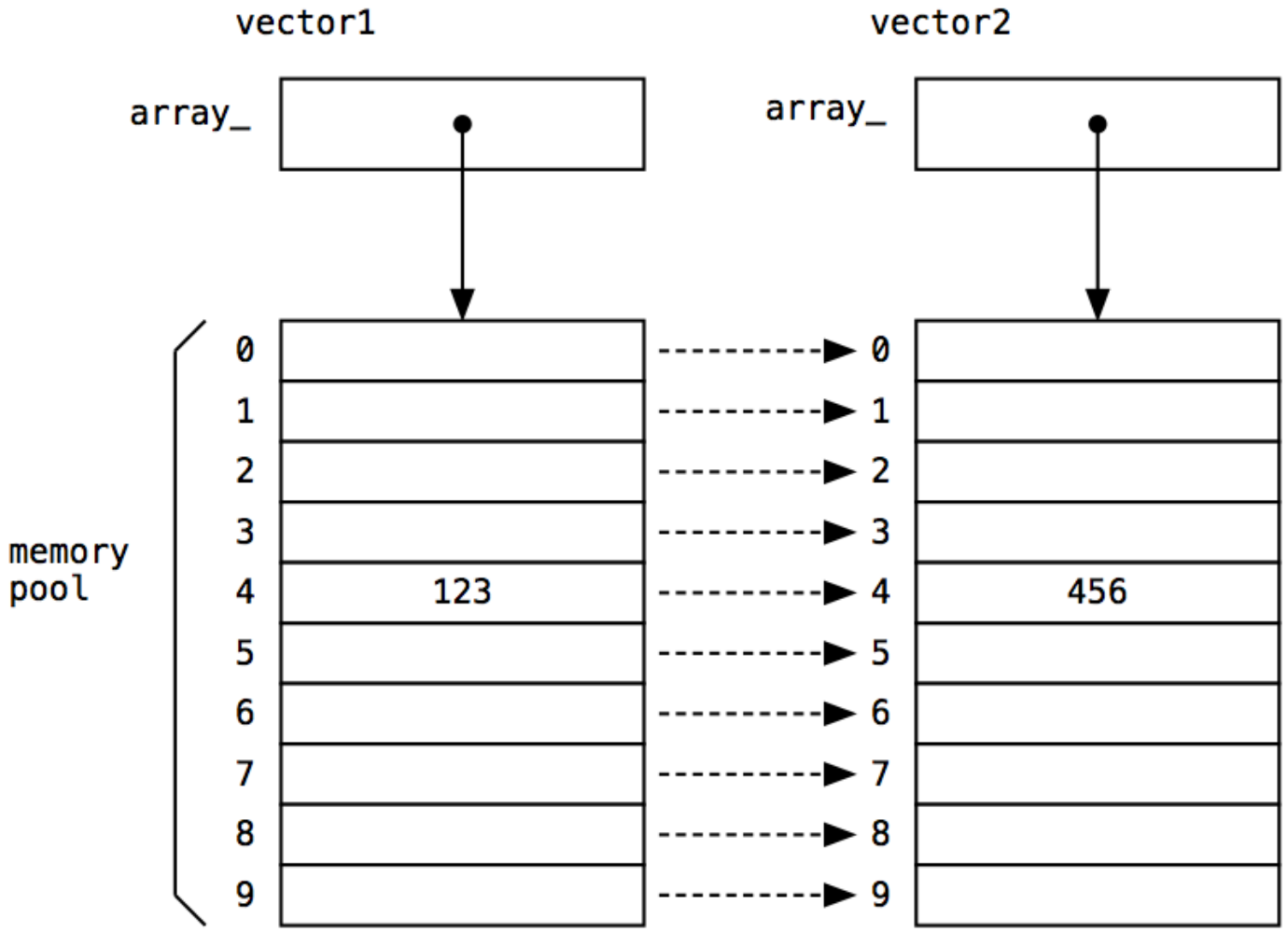
first delete and program crashes!



- vector2 needs its own copy of vector1
- Shallow copy of pointer only isn't enough
- Need deep copy of data that’s pointed to



vector2.Set(4, 456) leaves vector1 untouched



Deep Copy

```
FixedVector& FixedVector::operator=(const FixedVector& rhs)
{
	if (this != & rhs)
	{
		delete []array_;
		size_ = rhs.size_;
		array_ = new int[size_];
		for (int i = 0; i < size_; i++)
		{
			array_[i] = rhs.array_[i];
		}
	}
	return(*this);
}
```

Copy Constructor With Deep Copy

```
FixedVector     vector1;
FixedVector     vector2(vector1);


FixedVector::FixedVector(const FixedVector& a)
{
        size_ = a.size_;
        array_ = new int[size_];
        for (int i = 0; i < size_; ++i)
        {
                array_[i] = a.array_[i];
        }
}
```

The "Rule of Three"

If a class has a pointer data member, it must have:

1. A copy constructor that does a deep copy
2. An assignment operator function (operator=) that does a deep copy
3. A destructor that deletes memory allocated by *new*

It may be useful to have one private helper function that does the deep copy, which the copy constructor and assignment operator can call.