```
class Array {
    int* pData;
    const int size;

public:
    Array(int n) : size(n) {
        std::cout << "Allocate memory for array of size " << size << "\n";
        pData = new int[size];
    }
    ~Array() {
        std::cout << "Free array memory of size " << size << "\n";
        delete pData;
    }
    int& operator[](int i) {
        static int temp;

        if ((i < 0) || (i >= size)) {
            return temp;
        }

        return pData[i];
    }
};
```

```
int main() {
    Array arr1(5);
    Array arr2(arr1);

    arr1[0] = 1;
    arr2[0] = 2;

    std::cout << "1st element of arr1 = " << arr1[0] << "\n";
    std::cout << "1st element of arr2 = " << arr2[0] << "\n";

    return 0;
}
```

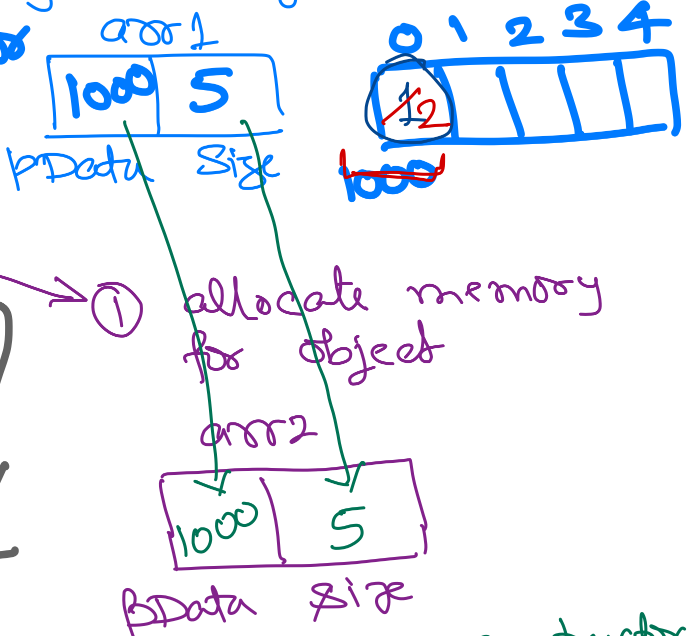
① allocate memory for object
② call constructor

arr1[0] = 1; → ref to 0th element of arr1's pData

arr2[0] = 2; → ref to 0th element of arr2's pData

2

2



① allocate memory for object
② call copy constructor (default)
⇓
makes copy of object by doing member wise copy.

Shallow copy

at function end
① call destructor arr2 ↳ free memory of addr stored in arr2's pData ⇒ 1000
② call destructor for arr1
⇓
free mem at 1000

```
class Array {
    int* pData;
    const int size;

    void copyArray(int *dest, int *src, int size) {
        for (int i = 0; i < size; ++i) {
            dest[i] = src[i];
        }
    }

public:
    Array(int n) : size(n) {
        std::cout << "Allocate memory for array of size " << size << "\n";
        pData = new int[size];
    }
    ~Array() {
        std::cout << "Free array memory of size " << size << "\n";
        delete pData;
    }
    int& operator[](int i) {
        static int temp;
        if ((i < 0) || (i >= size)) {
            return temp;
        }

        return pData[i];
    }
    Array(Array& obj) : size(obj.size) {
        std::cout << "Array copy constructor of size " << size << "\n";
        pData = new int[size];
        copyArray(pData, obj.pData, size);
    }
};
int main() {
    Array arr1(5);
    Array arr2(arr1);

    arr1[0] = 1;
    arr2[0] = 2;

    std::cout << "1st element of arr1 = " << arr1[0] << "\n";
    std::cout << "1st element of arr2 = " << arr2[0] << "\n";

    return 0;
}
```

copy constructor

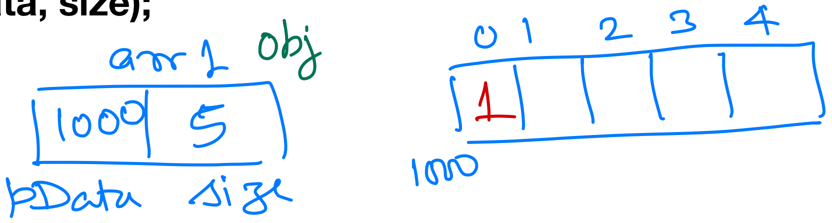
Copy constructor implementation doing deep copy

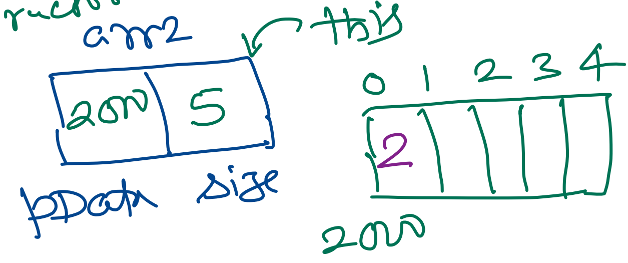
① allocate memory for arr2

② call copy constructor

2

1



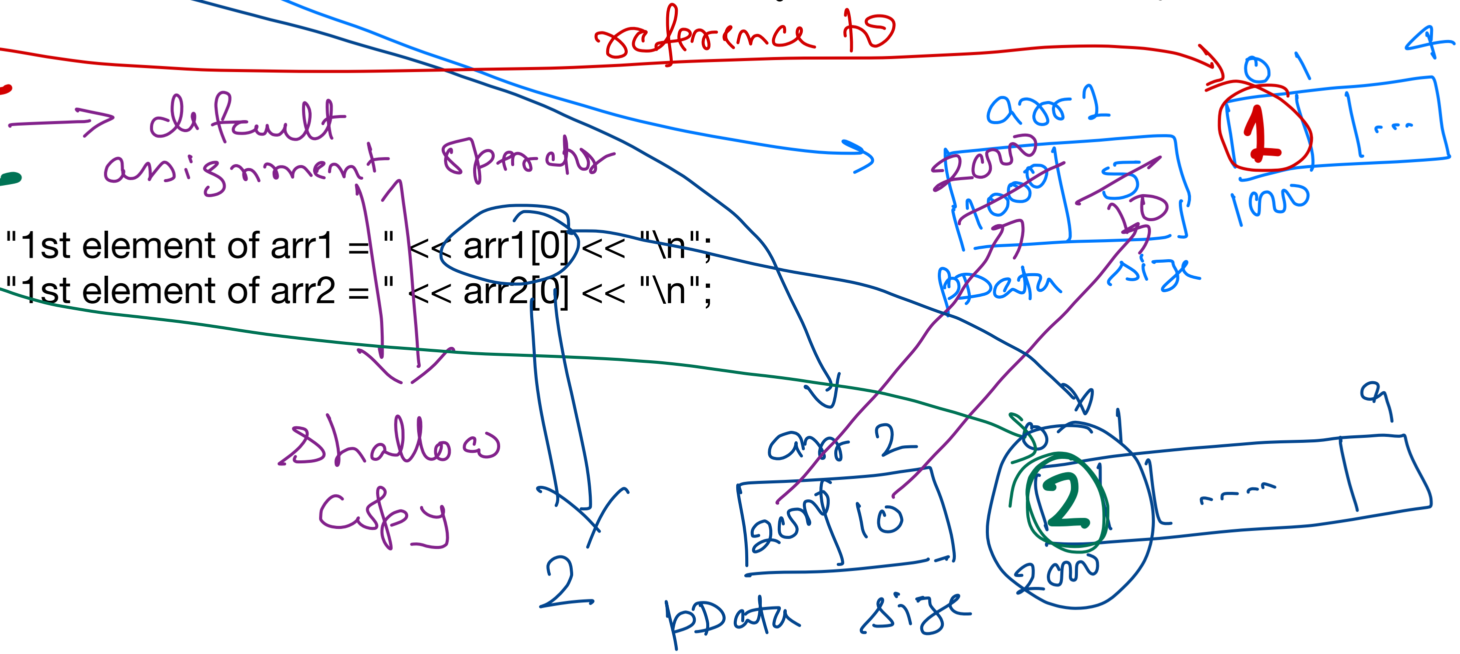


```
class Array {
    int* pData;
    int size;

    void copyArray(int *dest, int *src, int size) {
        for (int i = 0; i < size; ++i) {
            dest[i] = src[i];
        }
    }

public:
    Array(int n) : size(n) {
        std::cout << "Allocate memory for array of size " << size << "\n";
        pData = new int[size];
    }
    ~Array() {
        std::cout << "Free array memory of size " << size << "\n";
        delete pData;
    }
    int& operator[](int i) {
        static int temp;
        if ((i < 0) || (i >= size)) {
            return temp;
        }

        return pData[i];
    }
    Array(Array& obj) : size(obj.size) {
        std::cout << "Array copy constructor of size " << size << "\n";
        pData = new int[size];
        copyArray(pData, obj.pData, size);
    }

    Array operator=(Array& obj) {
        std::cout << "Assign array of size " << obj.size << " to array of size " << size  << "\n";
        size = obj.size;
        delete pData;

        pData = new int[size];
        copyArray(pData, obj.pData, size);

        return *this;
    }
};
```

deep copy in
assignment
operator

With default assignment operator impl.

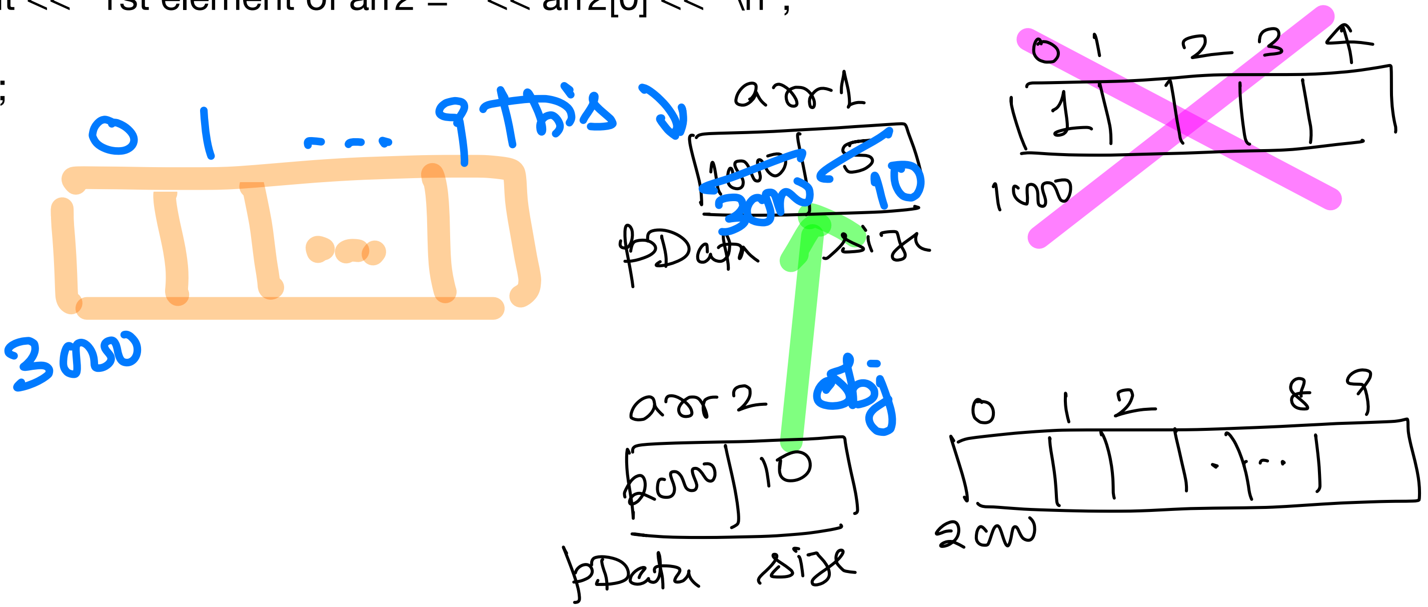
```
int main() {
    Array arr1(5);
    Array arr2(10);

    arr1[0] = 1;
    arr1 = arr2;
    arr2[0] = 2;

    std::cout << "1st element of arr1 = " << arr1[0] << "\n";
    std::cout << "1st element of arr2 = " << arr2[0] << "\n";

    return 0;
}
```



```
int main() {
    Array arr1(5);
    Array arr2(10);

    arr1[0] = 1;
    arr1 = arr2;
    arr2[0] = 2;

    std::cout << "1st element of arr1 = " << arr1[0] << "\n";
    std::cout << "1st element of arr2 = " << arr2[0] << "\n";

    return 0;
}
```

with assignment operator implemented to do deep copy.

arr1 = arr2; → arr1.operator=(arr2)



0 1 ... 9
this
arr1
~~1000~~ 3000
~~5~~ 10
pData size
0 1 2 3 4
1
1000
3000
obj
arr2
2000 10
pData size
0 1 2 8 9
2000