

Shallow Copy

Jyh-Shing Roger Jang (張智星)

CSIE Dept, National Taiwan University

Memory Allocation in Classes

- Common programming errors when using “new” for memory allocation in classes
 - Default **copy constructor** and **assignment operator** are based on “shallow copy”, which leads to errors easily.
 - We need to design our own constructor/operator.

```
class myVec {
public:
    myVec(int n){size=n; data=new int[size];};
    myVec(){size=10; data=new int[size];};
    ~myVec(){delete [] data;};
    int *data;
    int size;
};
```

Function overloading!

Double deletion!

Default **copy constructor** invoked
→ Shared memory & double deletion!

```
myVec a(100);
myVec b=a;
myVec c;
c=a;
```

Default **assignment operator** invoked
→ Share memory, memory leak, and double deletion!

How to Fix “Shallow Copy”?

- To fix the problem of “shallow copy”
 - Define our own copy constructor
 - Define our own assignment operator

```
// Copy constructor
myVec::myVec(const myVec& a) { // copy constructor from a
    size = a.size;           // copy sizes
    data = new int[size];     // allocate new array
    for (int i=0; i<size; i++) // copy the vector contents
        data[i]=a.data[i];
}

// Assignment operator
myVec& myVec::operator=(const myVec& a) { // assignment operator from a
    if (this != &a) {           // avoid self-assignment
        delete [] data;        // delete old array
        size = a.size;         // set new size
        data = new int[size];  // allocate new array
        for (int i=0; i<size; i++) // copy the vector contents
            data[i]=a.data[i];
    }
    return *this;
}
```

Avoid memory leak

Examples Which Fix “Shallow Copy”

○ Examples

- shallowCopy00.cpp: Demo of shallow copy
- deepCopy00.cpp: Use new copy constructor only
- deepCopy01.cpp: Use new assignment operator only
- deepCopy02.cpp: Use both

○ Lesson learned

- If a class allocates memory via “new” (or the likes, such as “malloc” or “calloc”), we should provide a new **copy constructor** and a new **assignment operator** to allocate new memory for the created copy.

Q & A

○ Questions

- How to avoid the error message (due to double deletion) in shallowCopy00.cpp?
- If we use STL vectors, do we still the problem of “shallow copy”? ➔ Please give examples and post to FB.

○ Further studies

- How to check memory leak?
 - Tools: Purify (Windows), Valgrind (Unix/Linux), Dr. Memory (both)
 - Please post on FB if you know other good tools to identify memory leak.
- How to avoid memory leak?
 - Use STL (standard template library)

Quiz

- A program is used to record each student's quiz scores

- Class definition:

```
class student {
public:
    student(int n=3){count=n; score=new int[count];};
    ~student(){delete [] score;};
    int *score, count;
    string name;
};
```

- Main program:

```
int main(){
    student a(3);
    a.name="John"; a.score[0]=70; a.score[1]=80;
    student b=a;
    b.name="Mary"; b.score[2]=90;
}
```

- Quiz:

- What are the contents of a and b?
- What are the two potential problems of this program?
 - Shared memory & double deletion