

# OBJECT MANIPULATION\*

Dr Duong Tuan Anh

This work is produced by The Connexions Project and licensed under the Creative Commons Attribution License †

## 1 1. OBJECTIVE

The objectives of Lab session 9 are (1) to learn to write parameterized constructor, constructor with default arguments and (2) to practice destructors.

## 2 2. EXPERIMENT

2.1 Run the following program in a C++ environment.

```
class Int{
private:
int idata;
public:
Int(){
idata=0;
cout<<"default constructor is called"<<endl;
}
Int(int d=9){
idata=d;
cout<<"constructor with argument is called"<<endl;
}
void showData(){
cout<<"value of idata: " << idata << endl;
}
};

void main()
{
Int i;
Int j(8);
Int k=10;
}
```

- Explain why the program incurs a compile-time error.
- Modify the program in order to remove the above error. Run the modified program.

- Run the following program in a C++ environment.

---

\*Version 1.1: Jul 7, 2009 1:06 am -0500

†<http://creativecommons.org/licenses/by/3.0/>

```

class Vector{
private:
int *value;
int dimension;
public:
Vector(int d=0){
dimension=d;
if (dimension==0)
value=NULL;
else{
value=new int[dimension];
for (int i=0; i<dimension; i++)
value[i]=0;
}
}
void showdata(){
for (int i=0; i<dimension; i++)
cout<<value[i];
cout<<endl;
}
~Vector(){
if (value!=NULL)
delete value;
}
};

void main()
{
Vector v(5);
v.showdata();
Vector v2(v);
v2.showdata();
}

```

- Explain why the program incurs one memory error at run-time.
- Now add the following code segment in the Vector class definition.

```

Vector(const Vector& v){
dimension = v.dimension;
value=new int[dimension];
for (int i=0; i<dimension; i++)
value[i]=v.value[i];
}

```

Check if the program still incurs the memory error or not. Explain why.

2.3 a. Test the following program in a Visual C++ environment:

```

class some{ // code segment a
public:
~some() {
cout<<"some's destructor"<<endl;
}
};

void main() {
some s;
}

```

```
s.~some();
}
```

What is the output of the above program during execution? Explain the output.

b. Test the following program in Visual C++ environment:

```
class some{ // code segment b
int *ptr;
public:
some(){
ptr= new int;
}
~some(){
cout<<"some's destructor"<<endl;
if (ptr!=NULL){
cout<<"delete heap memory"<<endl;
delete ptr;
}
}
};
```

```
void main()
{
some s;
// s.~some();
}
```

What is the output of the above program during execution? Explain the output.

c. In the main() function of the program in b, if we remove the two slashes (//) before the statement s.~some(); then what the result is when the program is executed ? Explain why.

2.4 Given the class definition as follows:

```
class Auto {
public:
Auto(char*, double);
displayAuto(char*, double);
private:
char* szCarMake;
double dCarEngine;
};
Auto::Auto(char* szMake, double dEngine){
szCarMake = new char[25];
strcpy(szCarMake, szMake);
dCarEngineSize = dCarEngine;
}
Auto::displayAuto(){
cout<< "The car make: " << szCarMake<< endl;
cout<< "The car engine size: " << dCarEngine<< endl;
}
void main(){
Auto oldCar("Chevy", 351);
Auto newCar(oldCar);
oldCar.displayAuto();
newCar.displayAuto();
}
```

1. Add an appropriate copy constructor to the Auto class .
  2. Add an appropriate destructor to the Auto class .
- c. Run all the modifications in a C++ environment.