

Lab 5: Constructors and Destructors

Objectives:

- To work with constructors and destructors
- To work with copy constructors

Part 1: Writing a Class with Constructors and a Destructor

In this part, you will define a class that models a barn on a farm.

Steps:

1. Start the Microsoft Developer Studio by clicking on the Windows *Start* menu: ***Start – Microsoft Visual C++ - Microsoft Visual C++.***
2. Create a new project for the lab by choosing *New* from the *File* menu and selecting the *Projects* tab. Select *Win32 Console Application* and then fill in:

Project Name: lab05
Platforms: Win32
Location: x:\cppclass\lab05 -- x: is where the labs are installed

Note: the *lab05* sub-directory should already exist. Make sure that “lab05” appears only once in the Location directory.

Then click on the *OK* button. Then select “An Empty Project” and press *Finish*, followed by *OK*.

3. Create a header file, BARN.H to contain the barn class definition. Define the following members (no inline functions):
 - A private integer that indicates how many cows are in the barn
 - A private *double* that contains the barn’s area
 - A private *char ** that contains the barn’s name
 - Four public constructors: one that accepts no arguments, one that accepts just a cow count, one that accepts a cow count and an area, and one that accepts a cow count, area and barn name.
 - A destructor
 - A *Display()* function

Be sure to put preprocessor directives in the header file to guard against multiple inclusion.

4. Create BARN.CPP and in it, implement the functions. The constructors should dynamically allocate an array of 256 characters for the barn name. The constructors should initialize the data members from the constructor arguments or set defaults. Use *strcpy* as required. The destructor should de-allocate the memory for the name. The Display function should display all of the data members to the console using *cout*.
5. Create CLIENT.CPP and write a *main* function. In main, dynamically create a barn using each of the four constructors. Call the display function and then free the objects. Test your program. Do you see the correct barn information?
6. Optional: repeat steps from an earlier lab to display any memory leaks to the console. Comment out the code in the destructor and retest – are there any memory leaks?

Part 2: Copy Constructors

In this part, you will learn why copy constructors are necessary.

Steps:

1. Add a new member function to the barn class: `void SetName (char *nameIn)`. In the new function, use *strcpy* to copy the input argument to the *name* data member.
2. In the client's main function, after the existing code (but before the *return* if you have one), insert these lines:


```
barn b5;
barn b6 = b5;
b5.SetName ( "Red Barn" );
b6.SetName ( "Black Barn" );
b5.Display();
b6.Display();
```
3. What do you expect the Display calls to print? Build and run your program. What do you see? (If you get an assertion, that's normal.) Why did this happen? Review the chapter on copy constructors and look at the barn class definition.
4. Write a copy constructor in the barn class and run the program again. Does it work now?

Optional: Assignment Operator

1. Add the following code to main after the existing code:

```
barn b7;
barn b8;
b8.SetName ( "Blue Barn" );
```

```
b7 = b8;  
b7.Display();  
b8.Display();
```

2. Build and run the program. What happens? Why doesn't the copy constructor run? The answer is that the copy constructor only runs when a new object is being created. Here we are simply assigning object to each other.
3. To fix this, you need to overload the assignment operator and do the same thing that you did in the copy constructor. For hints on overloading operators, refer to the appropriate chapter in the course notes.

