Constructors/Destructors for classes

```
1   using namespace std;
2   #include <fstream>
3   #include <iostream>
4   #include <string>
5   #include <iomanip>
6
7   class cityrecord
8   {private :
9           string city;
10          string state;
11          string zip;
12
13   public:
14      cityrecord ()  // Default Constructor
15          { city="Unknown";
16            state="Unknown";
17            zip="Unknown";
18          }
19
20      cityrecord (string temp) // Constructor for one string assumed to be the state
21
22          { city="Unknown";
23            state=temp;
24            zip="Unknown";
25          }
26      cityrecord (string tempstate,string tempcity) // Constructor for two strings assumed to be the state and city
27          { city=tempcity;
28            state=tempstate;
29            zip="Unknown";
30          }
31      cityrecord (string tempstate,string tempcity,string tempzip) // Constructor for three strings assumed to be the state, city and zip
32          { city=tempcity;
33            state=tempstate;
34            zip=tempzip;
35          }
36      ~cityrecord()  // Destructor
37      {cout << "\n\tInside destructor\n\n";
38      };
39
40      void printcity();  // Prints a single city on one line
41      void printform();   // Prints a single city inn a form vie
42  };
```

The default constructor initializes the city, state, and zip to value of ***Unknown*** since there are no parameters that have been passed to the constructor.

The other constructors are called in the same way that overloaded functions are designed. The number of parameters, the data type and order of the data types determines which one of these constructors have been activated.

This is the ***Destructor*** for the class ***cityrecord.*** This is used to return the memory used by the object ***city record*** for use by the system.

Arrays of classes

```
47  int main()  // Main program
48
49  { int i;
50    cityrecord city[4] = {
51                cityrecord(),
52                cityrecord("MS"),
53                cityrecord("MS","Gulfport"),
54                cityrecord("MS","Biloxi","39507")
55                };
56    cout <<"\n\n";
57    cityrecord *temp;
58    temp = new cityrecord("Moose", "Wiggins");
59    temp->printcity();
60    delete temp;
61
62    addcity();
63
64    printheader();
65    for (i=0;i<4;i++)
66       {
67         city[i].printcity();
68       };
69    cout<<"\n\n\tEnd of program\n\n";
70    return 0;
71  }
72
73  //---------------------------------------
74  void addcity()
75  {  cityrecord temp("MOO YORK");
76     temp.printcity();
77     return;
78  }
```

The array declaration calls the constructors and stores the data in the array as in this screen shot.

These are the address of the objects generated at run time. These will vary based according to the environment at the time of the program execution.

```
Wiggins         Moose      Unknown       0x3f6cf0
Inside destructor

Unknown         MOO YORK   Unknown       0x22fcd0
Inside destructor


City            State      Zip
-----------------------------------------
0   Unknown         Unknown    Unknown       0x22fd40
1   Unknown         MS         Unknown       0x22fd58
2   Gulfport        MS         Unknown       0x22fd70
3   Biloxi          MS         39507         0x22fd88

End of program

Inside destructor

Inside destructor

Inside destructor

Inside destructor
```

Notice that the destructor for each element of the array occurs as the program is terminating. Notice that the destructor messages for each array slot occurs just before the ***return 0; statement.***

Pointers to classes

These are the address of the objects generated at run time. These will vary based according to the environment at the time of the program execution.

The pointer declaration and the ***new*** operator calls the constructors and stores the data in the object as in this screen shot.

```
47  int main()  // (
48
49  { int i;
50    cityrecord city[4] = {
51                 cityrecord(),
52                 cityrecord("MS"),
53                 cityrecord("MS","Gulfport"),
54                 cityrecord("MS","Biloxi","39507")
55                 };
56    cout <<"\n\n";
57    cityrecord *temp;
58    temp = new cityrecord("Moose", "Wiggins");
59    temp->printcity();
60    delete temp;
61
62    addcity();
63
64    printheader();
65    for (i=0;i<4;i++)
66      {
67        city[i].printcity();
68      };
69    cout<<"\n\n\tEnd of program\n"
70    return 0;
71  }
72
73  //-----------------------------
74  void addcity()
75  {  cityrecord temp("MOO YORK");
76     temp.printcity();
77     return;
78  }
```

```
Wiggins         Moose      Unknown       0x3f6cf0
Inside destructor

Unknown         MOO YORK   Unknown       0x22fcd0
Inside destructor


     City            State      Zip
     ---------------------------------------
0    Unknown         Unknown    Unknown       0x22fd40
1    Unknown         MS         Unknown       0x22fd58
2    Gulfport        MS         Unknown       0x22fd70
3    Biloxi          MS         39507         0x22fd88

End of program

Inside destructor

Inside destructor

Inside destructor

Inside destructor
```

The ***delete*** operator destroys the object the pointer variable ***temp*** points to. Notice that the destructor has been invoked before the function ***addcity()*** has been called. The memory used by the object has been restored to be used by other processes.

Scope of classes in Functions

These are the address of the objects generated at run time. These will vary based according to the environment at the time of the program execution.

```
47  int main()  // Main program
48
49  { int i;
50    cityrecord city[4] = {
51                ci
52                ci
53                ci
54                ci
55                };
56    cout <<"\n\n";
57    cityrecord *temp;
58    temp = new cityrecord("Moose", "Wiggins");
59    temp->printcity();
60    delete temp;
61
62    addcity();
63
64    printheader();
65    for (i=0;i<4;i++)
66       {
67         city[i].printcity();
68       };
69    cout<<"\n\n\tEnd of program\n\n";
70    return 0;
71  }
72
73  //-----------------------------------------------
74  void addcity()
75  {  cityrecord temp("MOO YORK");
76     temp.printcity();
77     return;
78  }
```

```
      Wiggins         Moose      Unknown       0x3f6cf0

      Inside destructor

      Unknown         MOO YORK   Unknown       0x22fcd0

      Inside destructor


      City            State      Zip
      ------------------------------------------
0     Unknown         Unknown    Unknown       0x22fd40
1     Unknown         MS         Unknown       0x22fd58
2     Gulfport        MS         Unknown       0x22fd70
3     Biloxi          MS         39507         0x22fd88

      End of program

      Inside destructor

      Inside destructor

      Inside destructor

      Inside destructor
```

The function call ***addcity()*** will call the constructor for a local variable inside the function.

As the functions exits notice that the local variable ***temp*** destructor has been invoked to return memory used by the variable ***temp*** to the system.

As shown in the prior examples,

Constructors and ***destructors*** are for a class.

The classes can be :

- A standalone object as shown in the function ***addcity()***
- It can be in array of classes.
- The class can be used with pointers and therefore can be used for linked list or binary trees.

Constructor Errors		
C++ Statement	Error	Type of error
cityrecord temp2("MS","Biloxi","39507", "A");	Four parameters when there is not a matching constructor.	compile error
cityrecord temp2("MS","Biloxi",39507);	Zip code is sent in as a integer rather than a string.	compile error
cityrecord temp2("Biloxi","MS","39507");	Wrong order in parameter passing; Biloxi will be stored as the state and MS will be the city,	Logic error