struct Version of Date "Object"

```
struct Datetype
  {
   int day;
   int month;
   int year;
  };
```

structs are part of C and C++
Members of structs are data values

Class Version of Date "Object"

```
class Dateclass
  {
   public:
     int day;
     int month;
     int year;
     void outdate();
  };
```

We can now include "methods" (functions) in the declaration as well as data!!

NOTE: classes only available in C++ (not C)

Using the Date Class

```
#include "dateclass.h"

void main()
  {
   Dateclass today;
   .
   .
   today.month = 2;
   today.day = 27;
   today.year = 2001;
   today.outdate();
  } // End main
```

Methods are referenced just like other members

The outdate() Method

```
void Dateclass::outdate(void)
```

class name *class resolution operator*

```
{
 cout << month << '/' << day << '/'
      << year << endl;
} // END outdate
```

Other members of the class are available to all the methods of the class.

Information Hiding Version of Dateclass

```
class Dateclass
  {
   public:
    void outdate();
    int setdate(int, int, int);
   private:
    int day;
    int month;
    int year;
   };
```


Using the Date Class

```
#include "dateclass.h"

void main()
  {
   Dateclass today;
       .
       .
   today.month = 2;  // no longer legal!!


   today.setdate(2, 27, 2001);
       .
       .
   today.outdate();
  } // End main
```

Classes and Source Files

Source (code) files for classes can be organized in many ways. The following is a typical organization:

File 1 – dateclass.h – contains only the class declaration.

File 2 – dateclass.c – contains the definitions for all methods (functions) in the class.

File 3 – xxxxxx.c – contains the main program and all (non-class) functions.

Dateclass with Constructor

```
class Dateclass
  {
   public:
    void outdate();
    int setdate(int, int, int);
    Dateclass();
   private:
    int day;
    int month;
    int year;
  };
```

constructor method

Constructor Definition

```
Dateclass::Dateclass
  {
   day = 1;
   month = 1;
   year = 2001;
  }  // END Dataclass constructor
```

The constructor is like a function, except that it doesn't have a type.

Overloaded Constructors

```
class Dateclass
  {
   public:
    void outdate();
    int setdate(int, int, int);
    Dateclass();
    Dateclass(int, int, int);
   private:
    int day;
    int month;
    int year;
  };
```

Use of Constructors

```
int main()
  {
   Dateclass a(1,4,2001);

   Dateclass b;
        .
        .
        .
```

Uses 3-argument constructor

Uses "default" constructor (with no arguments)