Example – 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 are only available in C++ (not C) (In fact, C++ stands for "C plus Classes")

Using the Date Class

```
#include "dateclass.h"

void main()
  {
   Dateclass today;
   .
   .
   today.month = 2;
   today.day = 27;
   today.year = 1963;
   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;
   };
```

Now, the two methods (outdate and setdate) are available to the "outside world," but the data members are not

By default, ALL class members are private unless specified otherwise

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. Sometimes called the "interface" for the class.

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.

Typical File Organization for Classes

exampclass.h

```
class exampclass
  {
   public:
      void method1(int);
      int method2(char);
   private:
      int a, b, c;
      int method3();
  }  // END exampclass
```

- *Called the header or interface file*
- *Only contains class member declarations*

exampclass.cpp

```
#include "exampclass.h"

void exampclass::method1(int i)
  {
    // code for method1 goes here

  } // END method1


int exampclass::method2(char ch)
  {
    // code for method2 goes here

   return val1;
  }  // END method2

int exampclass::method3(void)
  {
    // code for method3 goes here

   return val;
  } // END method3
```

- *Contains method definitions*

main.cpp

```
#include "exampclass.h"
int main()
  {
   exampclass x, y;

   x.method1(n);
   k = y.method2(ch);
        .
        .
        .
  }  // END main
```

- *Contains method instantiations*

Compiling classes – Examples

To compile everything at once

```
g++ main.cpp exampclass.cpp
```

Separate compilation of class methods, then link with main

```
g++ -c exampclass.cpp
g++ main.c exampclass.o
```

Separate compilation of main and class, then create an executable named "example"

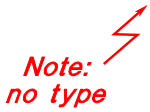
```
g++ -c exampclass.cpp
g++ -c main.cpp
g++ -o example main.o exampclass.o
```


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)