C++: Part 5
Classes and Objects

Vipin Bhatnagar
Dept. of Physics

M.Sc., M.Phil. Courses - 2007

Class & Object

CLASS

- An abstract data type
- Class definition doesn't eat any memory
- Once defined, it lives till the program terminates
- Class has unique name, attributes & methods
- eg.

Class name: ITEM
Attribute : Number, Price
Methods : GetNumber, DisplayCost

OBJECT

- Class variable
- Object creation occupies memory location
- Object can be created & destroyed at run-time
- Object has unique name, state & methods
- eg.

Object of class ITEM
object name : PEN
State : Number = 4, Price=2
Methods : GetNumber, etc.

Classes & Objects: Syntax

- Declare an object
  - `class-name object-name;`
- Declare an object with class definition

```
class class-name {
      ---------
   public:
      ---------
} object-name;        (*)
```

- Member Functions
- Member Types
  - Private members
  - Public members

Classes: Static Members

"Members declared inside a class BUT ***persisting*** from the declaration to the end of program"

Static members: both data as well as functions –

- ONE copy is shared b/w all objects
- Automatic initialization
- Visibility only within class lives till endofprog
- Used to maintain common values to class
- Defined outside the class

Syntax:

static data-type data-member-name;

    OR

data-type **static** data-member-name;

example:

```
#include <iostream>
//class declare
class DisplayCountClass {
  private:
    static int fncallDispCount; //static
  public:
    void DisplayCount(); };
```

contd..

```
//member function definition
void DisplayCountClass::DisplayCount()
{
  fncallDispCount++;    //increment count
  // display count
  cout<<"Number of times:"<<fncallDispCount<<"\n";
}

// static variable defined outside class
int DisplayCountClass::fncallDispCount;

int main(){
  DisplayCountClass Obj1;
  DisplayCountClass Obj2;  // two objects created
  Obj1.DisplayCount();
  Obj2.DisplayCount();     //display fn called
  Obj1.DisplayCount();     // fn called again
  return(0); }
```

Static Member Functions

- Accesses only static members of the class
- Syntax:

  static return-data-type function-name (arg list);
  OR
  return-data-type **static** function-name (arg list);
- Can be called with the class name if public
- Calling Syntax:

  class-name::member-function-name;

Classes & Objects: Part 6

Vipin Bhatnagar
Dept .of Physics

Static data member example...contd.

```
class DisplayCountClass {
  -----
  public:
    static void DisplayCount();  // static
       member fn
};
// define the member fn DisplayCount() as
   usual
int main()
{
  // create objects of class DisplayCountClass
  // add a call to the static member function
     now
  DisplayCountClass::DisplayCount();
  return (0);
}
```

Objects in Functions

- Passing objects as parameters (arguments)
  - Objects can be passed as value or as reference
  - Object passed as value can't be modified by fn
  - Fn directly works on the actual object if passed as reference

example:

(complex number and sum of 2 cmplx numbers:?)

```
#include <iostream>
// class declaration
class CmplxNum {
private:
  int RealAttribute;
  int ImgAttribute;     // two attribs
```

contd...

```
  Public:
    void AcceptAttribute();
    void DisplayAttribute();
    void GetSum(CmplxNum C1,CmplxNum C2);
};

// member function definition
void CmplxNum::AcceptAttribute() {
  cout<<“Enter real attributes  “;
  cin>>RealAttribute;
  cout<<”Enter imaginary attribute “;
  cin>>ImgAttribute;
}
void CmplxNum::DisplayAttribute() {
  cout<<RealAttribute<<” + ”
     <<”i”<<ImgAttribute<<”\n”;
}
```

eg. contd...

```
void CmplxNum::GetSum(CmplxNum c1,CmplxNum
   c2) {
RealAttribute = c1.RealAttribute +
   c2.RealAttribute;
ImgAttribute  = c1.ImgAttribute +
   c2.ImgAttribute;
}
// main function
int main()
{
  CmplxNum C1; CmplxNum C2; CmplxNum C3;
  // readin inputs
  C1.AcceptAttribute();
  C2.AcceptAttribute();
  // get sum of C1 and C2 by passing as value
  C3.GetSum(C1,C2);
  C3.DisplayAttribute(); return(0); }
```

Passing Objects by Reference

- mphil
  - ls: vipin
  - cd vipin; ls: frile1.C, file2.C
  - cd ..
  - cp vipin/file1.C newfile.C
  - pwd: /home/mphil
  - mv newfile.C vipin/newnewfile.C
  - cd vipin; cd cplus;
  - cp ~/newfile.C .

Mathematical Operators

- Simple Ones (directly availale)
  - Assignment int a = 5;
  - Arithmetic operaors
    - +
    - -
    - *   (multiplication)
    - /   (division)
    - % (modulo, int a = 17%5 will give a = 2)
  - Compound Assignment
    - += (a+=2; is a=a+2) **NB**: i++ is i = i+1 suffix/prefix
    - -=   (a -=4; is a=a-4)   i-- is i = i-1
    - *= (a *= b; is a = a*b)
    - /= (a /= b; is a = a/b)

Math ops contd.

  - Prefix: b = 2; a =++b; ans is a=3, b =3
  - Suffix: b = 2; a=b++; ans is a=2, b = 3
- #include <math>
  - Gives you access to:
    - cos,sin,tan,acos,..
    - cosh,sinh,tanh,exp,log,log10
    - pow requires two args, eg. “7^2 is pow(7,2) = 49
    - sqrt