University of Engineering and Technology,
Peshawar, Pakistan

CSE 102: Computer Programming

Lecture 8

Objects and Classes

By;
Dr. Muhammad Athar Javed Sethi

Object Oriented Programming (OOP)

- The most import feature of C++ programming is that it supports Object Oriented Programming (OOP).
- In OOP computer program is divided into objects.
- OOP language is easy and flexible approach for designing and organizing the program.
- The program is designed using classes.

Class

- Class is a collection of data and functions.
- The data items and functions are defined with in the class.
- The functions are written to work upon the data items and each function has a unique relationship with the data item of the class.

Class *(cont)*

- Classes are defined to create user define data types.
- These are similar to built in data types available in all programming languages.
- Definition of a data type does not create any space in the computer memory, while when a variable of that data type is declared, a memory space is reserved for that variable.
- Similarly when a class is defined, it does not occupy any space in the computer memory, it only defines the data item and the member function that can be used to work upon its data items.

Class *(cont)*

- Define a class only specifies its data members and the relationship between the data items through its functions.
- Classes and structure are similar. The syntax of a structure and class is also similar.
- Generally, structures are exclusively used to hold data while classes are used to hold both data and functions.

Defining a Class

- Class is defined in a similar way as a structure.
- Keyword “class” is used to define the class.

Syntax;

```
class class_name
{
   body of the class
};
```

Members of Class

- Class contains data items and functions, these are called members of the class.
- The data items are called Data Members.
- The functions are called Member Functions (methods).

Data Members

- The data items of a class are called data members of the class.

```
class abc
{
int w,x,y,z;
float a,b;
};
```

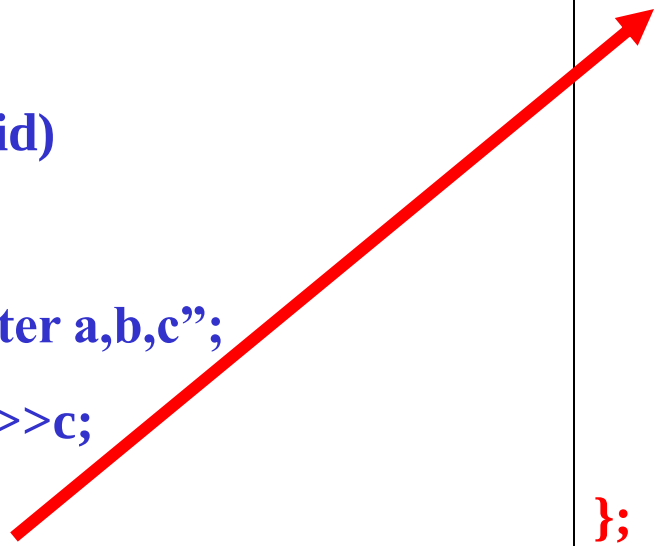
Summary: In the above class a,b,w,x,y and z are data members of the class “abc”

Member Functions

- The functions of a class that are defined to work on its data members are called member functions of that class.
- The member function may be defined with in the class or outside it.

Member Functions

```
class abc
{
int a,b,c;
      void get(void)
      {
      cout<<"Enter a,b,c";
      cin>>a>>b>>c;
      }
      void pout(void)
      {
      cout<<"a=" << a << endl;
      cout<<"b=" << b << endl;
      cout<<"c=" << c << endl;
      }
};
```



Summary: In the above class there are three data members and two member functions. The member functions are "get" and "pout". The get functions is used to input values into data members a,b,c. The pout functions is used to print values of the data members on the computer screen

Member Access Specifier

- The command that determines whether a member of a class can be accessed from outside the class or not are called member access specifiers.
- Normally two types of access specifiers are used;
  - Private:
  - Public:

Member Access Specifier *(cont)*

- **Private:**
  - The members of a class that can be accessed only from with in the class are called private members.
  - They can not be accessed from outside that class.
  - Normally all data members are declared as private.
  - The members functions can also be declared as private.
  - All class members that comes after the specifier private and upto the next member access specifier are declared as private and are accessible only from inside the class.
  - The default access mode is private, thus if no access specifier is given, the members are treated as private.
  - Making a data to be accessed from with in the class is called data hiding (Data Encapsulation) i.e. the data declared as private is hidden from outside the class.

Member Access Specifier *(cont)*

- **Public:**
  - Public members of a class can be accessed both from inside and from outside the class.
  - Normally, member functions are declared as public.
  - The data members can also be declared as public.

Member Access Specifiers

```
class cdate
{
private:
int y,m,d;
public:
      void gdate(void)
      {
      cout<<“Year?”; cin >> y;
      cout<<“Month?”; cin >> m;
      cout<<“Day?”; cin >> d;
      }
      void pdate(void)
      {
      cout<<d <<“/”<<m<<“/”<<y endl;
      }
};
```

Summary: In the above class “cdate” there are three data members y,m,d of int type and declared as private. While two member functions “gdate” and “pdate” are declared as public.

Variable Vs Objects

Variable

- A data type is used to declare a variable.
- A variable of a data type is also known as the instance or a case of that data type.
- Each variable has a unique name but each variable follows the rules of its data type.
- When a variable of a data type is declared, some space is reserved for it in the memory.

Object

- Class is also like a data type.
- It is therefore used to declared variables or instances.
- The variables or instances of a class are called objects.
- Class may contain several data items and functions.
- Thus the object of a class contain all data items and functions(of that class).

Objects

- Objects represents data members of a class in the memory.
- Each object of a class has a unique name
- The name of the object differentiates it from the other objects of the same class.
- The values of data members of different objects of the same class may be different or same.
- The values of the data members in an object are known as the state of the object.

Declaring Objects of a Class

- The objects of a class is declared in a similar way as the variables of any data or structure type are declared.
- When a class is define no space is reserved for it in the memory. It only provides information how its objects will look like.
- When an objects of a class is declared, a memory space is reserved for that object.
- The syntax to create objects of a class type is;

Syntax;

class_name object_name;

Example

- cdate aniv;
- sum myaverage;
- sum myaverage,prvaverage;

Calling Member Functions

- The member functions of a class is called in a similar way as a member or data item of a structure is called.
- The member function is called through an object of the class.
- The dot operator is used.
- The dot operator connects the object name and the member function.
- The dot operator is also called the class member access operator.

Syntax;

object_name.member[with arguments in case of a function]

Example

- add.pdate( );

Here add is the name of the object and pdate is the member function.

Defining Member Functions Outside the Class

- Member functions can also be defined outside the class.
- In this case only the prototype of the member function is declared inside the class.
- The member functions are defined outside the class in the similar way as user defined functions are defined.
- However, the scope resolution operator (::) is used in the member function declarator to define the function of class outside the class.
- Member function is define outside the class only if body of the function definition is large, otherwise function definition should be defined inside the class.

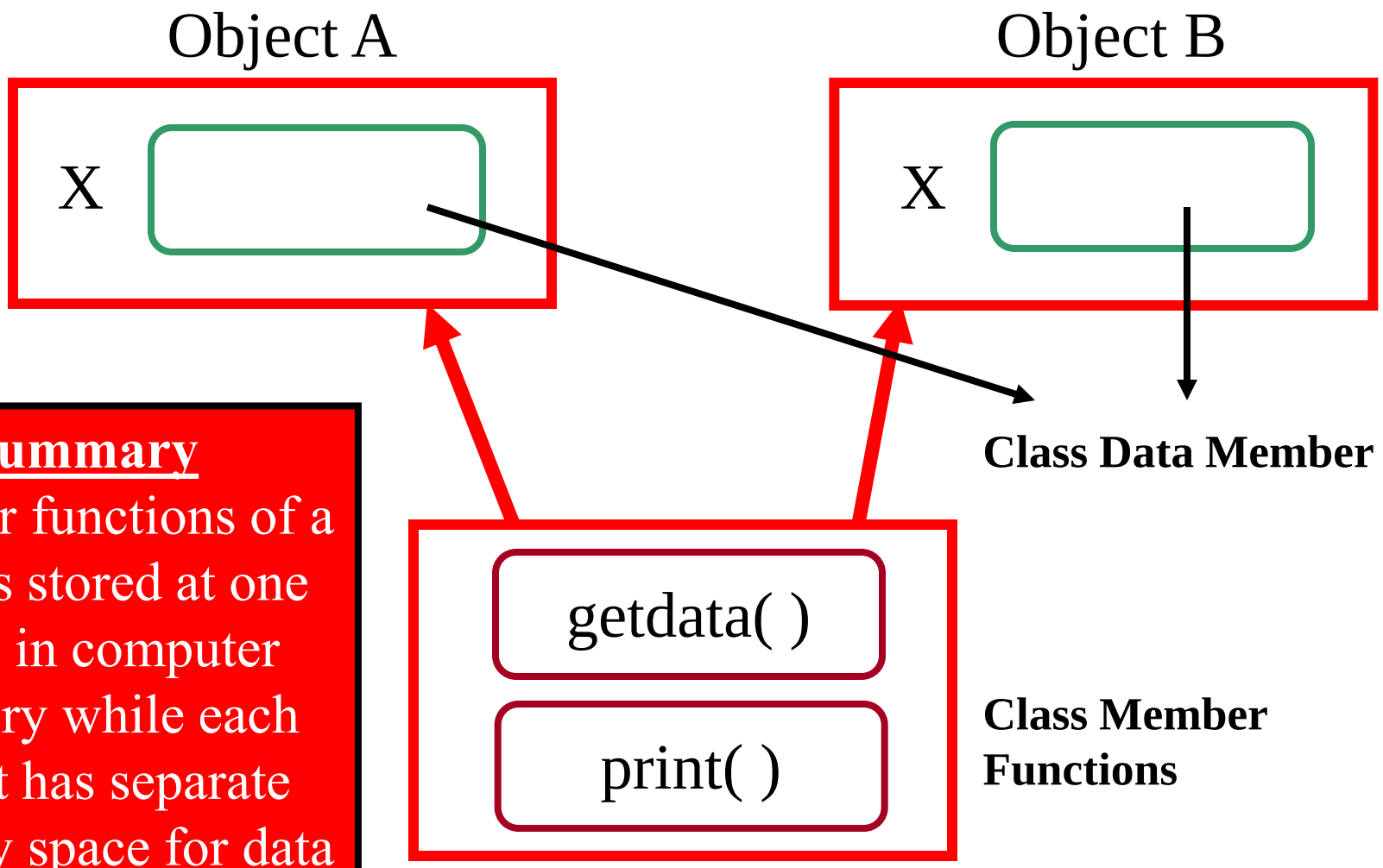
Syntax;

```
type class_name :: function_name (arguments)
{
   body of function
}
```

Storage of Object in Memory

- When object of class is created, a space is reserved in the computer memory to hold its data members.
- Similarly, separate memory spaces are reserved for each class object.
- Member functions of class are stored at only one place in the computer memory, all objects of the class use the same member functions to process data.

Storage of Objects in Memory



Summary

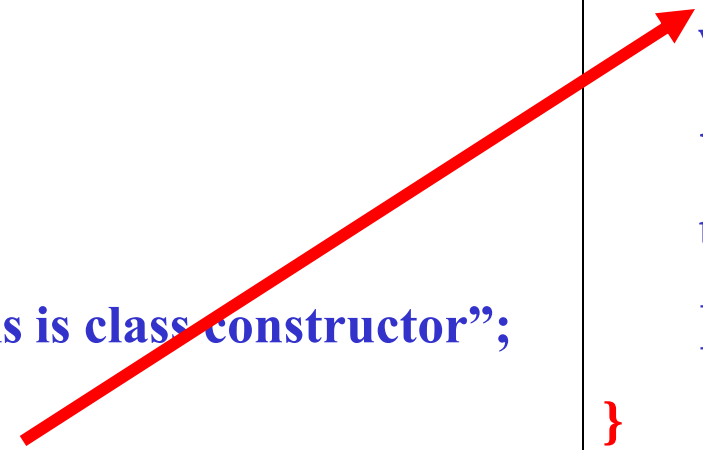
Member functions of a class is stored at one place in computer memory while each object has separate memory space for data members

Constructors

- A constructor is a member function of a class.
- It is executed automatically when an object of that class is created.
- The name of the constructor function is the same as the name of the class itself.
- Constructor function may have arguments but it can not return any value.

Constructor *(cont)*

```
#include <iostream.h>
class test
{
    public:
    test( )
    {
    cout<<"this is class constructor";
    }
void main(void)
    {
    test a,b,c;
    }
}
```



Summary: In the above program three objects a,b,c of the class test are created. Each time an object of the class "test" is created, the constructor is called and "this is constructor" is printed. Since three objects are created, the statement "this is constructor" is printed three times.

Initializing Data using Constructors

- The constructor functions are normally used to initialize values in the data members of a class when the program is executed, this type of initializations is called the automatic initialization.

Destructors (Function)

- When an object is destroyed, a special member function of that class is executed automatically, this member function is called Destructor.
- The destructor functions has the same name as the name of a class but a tilde sign (~) is written before its name.
- It is executed automatically when an object comes to the end of its life.
- Like constructors, destructors do not return any value.
- Destructors do not take any arguments.
- Destructor functions are commonly used to free the memory that was allocated for objects.

Life Time Of Objects

- Local Object
  - Local object is destroyed when all the statements of the function in which it is declared are executed. So at the end of the function, the destructor function is executed.
- Global Object
  - Global objects are declared before main function.
  - Global objects or static objects are destroyed when program execution ends.
  - So at the end of the program the destructor function is executed.

Passing Objects as Arguments

- When object is passed as an argument to a member function
  - Only the name of the object is written in the argument.
  - The number of parameters and their types must be defined in the member function to which the object is to be passed. The object that are passed are treated local for the member function and are destroyed when the control returns to the calling function.