23/08/2023

Unit II: Class & Objects
(Weightage - 18 marks)

Class

Questions

Q1 Define Class with syntax (2M)

Q2 WAP to declare a class student having data members as stud_id name and roll no. ~~accept and dec~~ accept and display for 5 student.

Q3 WAP to declare a class mobile having data members as prices and model number. Accept and display the data for 10 objects;

Q4 WAP to declare a class student the data members of
Roll no,
name
marks.

Q5 WAP to declare class circle with data members as
radius
area.
Declare a function get data to accept radius and put data to calculate and display area of circle.

Page No.
Date

Qb. WAP to declare a class 'Account' with data members as acc no, name and bal. Accept data for eight acc and display details of accounts having balance less than 10000. (6 marks)

Q1. Define class with syntax.

⇒ A class in c++ is an user defined datatype that binds data and functions that operates on data together in a single unit.

Generally, a class specifications has two parts
class declaration - shows its data members.
class functions definitions - shows member function.

Syntax

```
class class_name
{
  private :
        variable declaration;
        function declaration;
  public :
        variable declaration;
        function declaration;
};
```

// body of the class.

24/08/2023

WAP to declare a class student of data members of
Rollno. name marks.

```
#include<iostream>
using namespace std;
class Student
{
int rollno;
char name[50];
int marks;
public:
void getdata()
{
  rollno = 10;
  name = "xyz";
  marks = 100;
}
void display()
{
cout << "Roll number is " << rollno;
cout << "Name : " << name;
cout << "Marks " << marks;
}
};
void main
{
student s;
 s.getdata();
s.display();
return 0;
}
```

WAP to declare a class student having data members as stud_id name and roll no. Accept and display for 5 students

```
#include<iostream>
using Namespace std;
class Student
{
int roll_no;
char stud_name[50];

public:
void getdata()
{
 cout<<"Enter name & rollno";
 cin >> stud_name >> roll_no;
}
void display()
{
cout<< stud_name << roll_no ;
}
void main
};
 int main()
{
 Student s[5];
 for(int i=0; i<5; i++
  {
   s[i].getdata();
  }
cout <<"student details:";
for(int i=0; i<5; i++)
  {
   s[i].display();
  }
 return 0;
}
```

08/23
Q3

WAP to declare a class mobile having data members price and modal number. Accept a data for 10 objects.

```
#include<iostream>
using namespace std;
class mobile
{
int price;
int modal_number;
public:
void getdata()
{
cout<<"Enter price";
cin>>price;
cout<<"Enter modal_no";
cin>>modal_number;
}
void display()
{
cout<<price<<modal_number;
}
void main()
{
price p[10];
for(int i=0;i<10;i++)
{
p[i].getdata();
}
cout<<"mobile details";
for(int i=0;i<10;i++)
{
p[i].display();
}
}
```

2231C

Q5 WAP program to declare class with data members - radius, area. Declare a fun to get data to accept radius and to calculate and display area of circle.

⇒
```
#include <iostream>
using namespace std;
class circle
{
float radius;
float area;
public:
void getdata()
{
  cout<<"Enter radius";
  cin>>radius;
}
void putdata()
{
    area = 3.14 * radius * radius;
    cout<<"Area of circle = "<<area;
}
};
int main()
{
circle c;
c.getdata;
c.putdata;
return 0;
}
```

```
Enter radius: 2
Area of circle: 12.56
```

06 WAP to declare a class Account with data members acc_no, name and bal. Accept for 3 and display <100

```
#include <iostream>
using Namespace std;
class Account
{
  int acc_no[50], [50];
  char name[50];
  int bal;
  public:
class Account
{
  long int accno, bal;
  char name[50];
  public:
void getdata()
{
cout<<"Enter account number, balance, name";
cin>>acc no>>bal>>name;
}
void putdata()
{
if(bal<10000)
  {
  cout<<"Account no: "<<accno;
  cout<<"Balance : "<<bal;
  cout<<"Name : "<<name;
}
}
};
```

2231C

```
int main()
{
Account a[8];          //object.
for (int i=0; i<8; i++)
   {
     a[i].getdata();
   }
for(int i=0; i<8; i++)
   {
     a[i].putdata();
   }
return 0;
}
```

26/08/22

Q. Access specifiers with suitable examples. (4ms)

- Access specifiers are use to restricts accesibility of of class members.
- Access specifiers of class defines how the members of the class that are data members and member fun^n can be access.
- Each member of the class is associated with excess specifiers.
- It controls its accessibity as well as determines the part of the program that can directly access member of the class.
- In the class declaration syntax the keywords private, public and protected are known as ~~modes~~. access specifiers also known as visibility ~~boonds~~.
- When members declared private it can be access inside the class, while a public member accessible both inside and outside the class.
- Protected members are accessible both inside the class in which they are declared as well as

inside the derived (child) classes of the same class.

- Access specifiers are provided by writing appropriate keyword (public, private, protected) followed by a colon :
Note that the default access specifier is private that is no access specifier is provided it is considered to be private.

The general syntax of class declaration that uses private and public access specifiers that are given below.

```
class classname ()
{
private :
 //Declaration of private
 // class member (data member & member function)
public :
 // Declaration of public
 // Class members
};
```

Example

```
class Book
{
 Private :
 char title [30];
 float price ;
 public :
  void getdata ();
  void putdata ();
};
```

22316

Page No.
Date

In above example, class name book with two data members title and price and member function void getdata() & putdata() This Data members are declared as private & member function are declared as public it implies that data member are accessable by through member function while the member function are accessable anywhere in program.

Defining data member and member function inside the class and outside the class.

Write general form of member function. (Defination, out of the class) 2m

OR

How user can declare member function outside the class.

Give syntax of defining a member function inside and nesting of ~~mem~~ of fun in the class with example (4m)

→ Define class members inside the class.

→ When a function is defined inside a class it is treated as inline function.

Syntax

```
class class_name
{
  access specifier :
  data member ;
  access specifier :
  return_type fun_name()
  {
    body of function
  }
};
```

Example

```
class student
{
int roll_no;
char name[20];
public:
void getdata()
{
cin >> roll_no >> name;
}
void putdata()
{
cout << roll_no << name
}
};
```

22316

25/08/2023

Page No.
Date

Define class members outside the class

⇒ A member function of a class can also be defined outside the class.

It requires the function declaration to be provided inside the class definition.

The member function in C++ is declared inside the class like a normal function.

This declaration informs the compiler that the function is member of the class and it has been defined outside the class.

The definition of member function outside the class has function name in function header is preceded by class name and scope resolution operator.

Syntax

```
return type  class name :: function_name()
{
// body function.
}
```

Example

```
class student
{
  ---------
  ^^^^^^^^^

public :
  void getdata(); // declaration of memberfun
};
void student :: getdata()
{
  ---------
  ---------
}
```

Scanned with CamScanner

imp

Memory allocation of objects.

Q1 Describe memory allocation for objects.

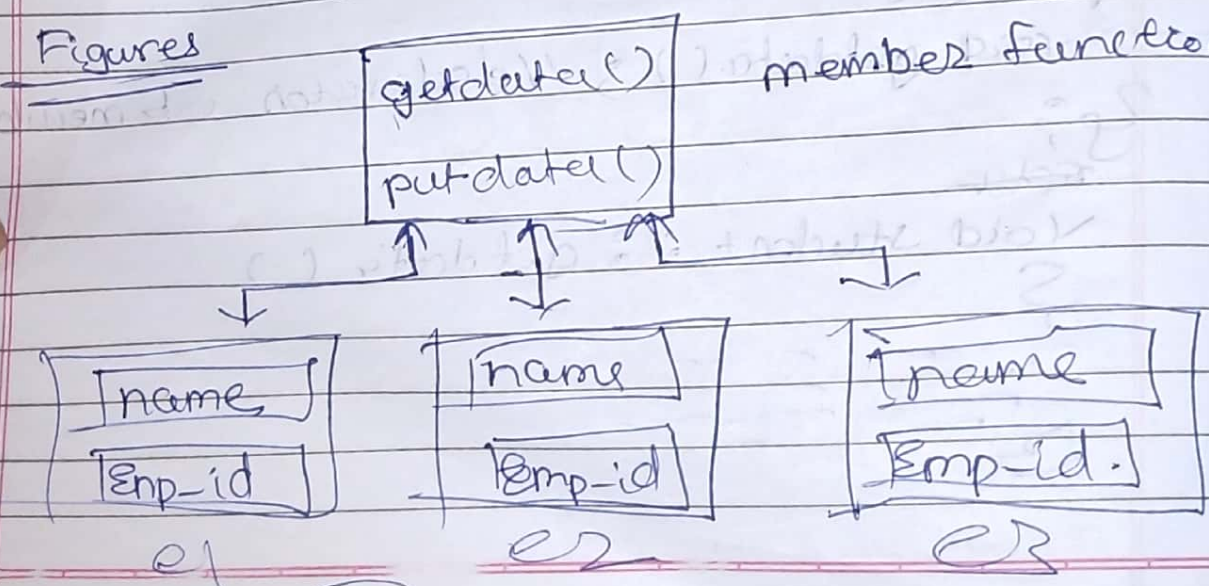
OR

Q2 Describe how memory allocated to object of class with suitable diagram.

⇒ Once we defined class it will not allocate memory space for the data member of the class.

* The memory allocation for the data members of the class is performed seperately each type when an object of the class is created.
* When an object of an class is created some amount of memory is allocated to it.
* The amount of memory allocated to an object is equal to the amount of memory required by data members of the class.
* This is because each object of an class has it own copy of data members.
* All the objects of class share some copy of member functions of a class.

Figures



2231C

* In the figure, member functions getdata() & putdata() are shared by all the three objects e1, e2, e3 and each object has it's own copy of data members of class.
* Once the object of an class is created it needs to be initalized valid values before it is used in the program.
* The object - To initalize object of class member function of same class is to be defined.
* It will assigned valid values to the data members of an class.
* However, it requires explict call to member function for every object.
* The memory allocation to the object is done by Speical member function that is constructor.
* The deallocation of the memory is done by another speical function theat is destructor

STATIC DATA MEMBERS
STATIC MEMBERS FUNCTIONS
FRIEND FUNCTION.

Page No.
Date

Questions

Q Explain with suitable example FRIEND function. (2m)

Q Write any two characteristics of friend function. (2m)

Q// Write two properties of static member function.

Q// Explain Friend function with proper example.

Q// Describe use of static Data member 2m/4m.

Q// WAP to greatest no among two numbers from two diff classes using friend fun^n. (6m)

Q// WAP to declare two classes with data members as m_1 & m_2 use friend function to calculate average of two (m_1, m_2) marks & display it.

Q// Write any two characteristics of static data member.
WAP to count no. of object created with the help of static data mut

2231c

Page No.
Date

Static Data members

① In c++ data member of a class can be qualified as static.
② Static variables are normally used to maintain values common to the entire class.
③ Static Data members is useful when all objects of the same class must share common information.
④ Static Data members are used to share information among the objects of class.
⑤ Syntax
static data type data-member-name;
or
datatype static data-members-name;

⑥ Example
static int a;
int static a;

Syntax
data-type class-name :: static data member = value;

Example
int student :: static roll-no = 10;

Properties of static Data members
or
Characteristics of static Data members.

Fixed

① All static variables are automatically initialize to zero when the first object of class is created.
② Only one copy of static Data member is created for the entire class and is shared by all the objects of that class.
③ It is visible within the only class but it's life time is the entire program.
④ Static Data members are normally used to maintain values common for all objects.
⑤ The type and scope of each member must be redefined outside the class definition because static data members are stored seperately

one data member share

Scanned with CamScanner

rather than as a part of an object.

⑥

28/09/2022

Use of Static Data members

intmain()

① Only one copy of that member is created for the entire class

② Only one copy is shared by all the objects of that class, No matter how many objects are created.

③ It is initialize before any object of this class is created even before the main starts.

④ It is visible only within the class but it's lifetime is the entire program.

⑤ Typical use of static member is for recording data common to all objects of a class.

Q WAP of ~~C++~~ to count where no. of object

Q WAP to count no. of object created with the help of static data members.

Example of static member function ⟹

22316

```
#include <iostream>
using namespace std;
class test
{
 int objNo;
 static int count;
public:
 void test ()
{
 objNo = ++count;
}
 void ~ ()
{
 --count;
}
void printobjNum ()
{
cout << "object Number : " << objNo;
}
static void printobjcount ()
{
 cout << "count is " << count;
}
};
int main()
{
 test t1, t2;
test:: printobjcount ();
test t3;
test:: printobjcount ();
t1. printobjNumber();
t2. printobjNumber();
t3. printobjNumber();
return 0;
}
```

Scanned with CamScanner

Output

```
Count : 2
Count : 3

Object Number : 1
Object Number : 2
Object Number : 3
```

Static member function

* Member function may also be static.
* A static function can have access to only other static members declared in the same class but there are following restrictions on them.
  1. They may only access other static members of the same class directly.
  2. They do not have this pointer.
  3. There cannot be static and non-static version of the same function.
* Static member functions can be called using class name as,
  class name :: function name();
* Syntax declaration of static member fun^n.

```
static return-type function name();
  OR
return-type static function name();
```

22316

* For example:
  Static void getdata();
  OP
  void Static getdata();
* Example of calling static member function:

```
Abc::getdata();
```

Q// WAP to count no. of object created with the help of static data members & Static member functions

```
#include<iostream>
using namespace std;
class counter
{
private:
static int count;
public:
counter()          // in constructor increase the count values
{
count++;
}
void display()
{
cout<<"the number of object created"<<count;
}
};
int counter::count=0;
int main()
{
counter c1;
counter c2;
counter c3;
c3.display();
return 0;
}
```

Output
the number of object created: 3

FRIEND FUNCTION

* A FRIENT function of a class is defined outside that class but it has right to access all private and protected members of the class.
* Even though the prototype for friend functions appear in the class definition, (friends are not member function)
* A friend can be function or class in which the entire class and all of its members are friends.
* To declare a function as a friend of a class precede the function prototype in a class definition with keyword friend.
* Syntax

friend return type function name();

* Example

friend void getdata();

Scope

* A friend function in c++, have following properties.

(1) The friend function is not in the scope of the class in which it has been declared. (body = scope of fun)

(2) It cannot be called using the object of that class.

(3) It can be invoked directly like a normal function.

(4) It cannot access data members directly. This are accessed with object name and . dot membership operator.

Scanned with CamScanner

22316

Page No.
Date

&amp obj1.m;

⑤ Friend can be declare anywhere. (in public, private, protected in any section in the class) without changing its meaning.

⑥ Friend function cannot be inherited.

pg 23

* WAP to declare two classes with data members as m1 & m2 use friend function to calculate average of two (m1, m2) marks & display it. -6m

⇒

```
#include <iostream>
using namespace std; class class_2;
class class_1
{
  int m1;
public:
  void getdata()
  {
    cout<<"Enter m1";
    cin>>m1;
  }
friend void average(class_1 p, class_2 q);
};
class class_2
{
  int m2;
public:
  void getdata()
  {
    cout<<"enter m2"
    cin>>m2
friend void average(class_1 p, class_2 q);
};
```

```
void average (class_1 p, class_2 q)
{
    float avg = (p.m1 + q.m2)/2.0;
    cout << "Avereege = " << avg;
}
int main()
{
    class_1 c1;
    class_2 c2;
    c1.getdata();
    c2.getdata();
    average(c1, c2)
}
```

Output

```
Enter m1    10
Enter m2    20
Average = 10
```

22316

13/03/23

Page No.
Date

Find to greatest among two numbers. Using friend

```
#include<iostream>
using namespace std;
class class_2;
class class_1
{
  int a;
pubic:
  void getdata()
  {
    cout<<"Enter 1st Number";
    cin>>a;
  }
friend void greatest(class_1 p, class_2 q);
};
class class_2
{
  int b;
public:
  void getdata()
  {
    cout<<"Enter 2nd Number";
    cin>>b;
  }
friend void greatest(class_1 p, class_2 q);
};
void greatest(class_1 p, class_2 q)
{
  if(p.a > q.b)
    {
      cout<<"Number 1 is greatest";
    }
else
  {
  cout<<"Number 2 is greatest";
  }
```

Scanned with CamScanner

```
Void main()
{
 class_1   x1 ;
 class__2  x2 ;
 x1.getdata();
 x2.getdata();
 greatest(x1,x2)
}
```

Output

```
Enter 1st number 69
Enter 2nd number 51
Number 1 is greatest number
```

Page No.
Date

Constructors

Questions

* Define constructor, list types of constructor
* Write a C++ program declare class student with data member roll no and name declare a constructor to initalize data members of class
* Describe constructor with default arugement with an example.
* What is parameterized constructor?
* Describe with example passing parameter to base class constructor and derived class constructor by creating object of derived class (4m)
* Differiate between Constructor & Destructor! (4m)
* write a C++ program to declare a class addition with data members x and y with constructor. Calculate addition and display it using function. (display).
* WAP to declare a class student with members as roll no, name and department. Declare a pareterised constructor with default value for department as CO to initalize members of object. Initalize and display data for two student. (6marks)
* WAP for constructor with default argument & use of destructor. (6m)

casting

Q1 Define Constructor. Enlist its types?

⇒ * A constructor is a special member function which is used to allocate memory space and values to data members of that object.
* Constructor makes the object function by converting an object with unused memory into useable object
* whenever an object is created the constructor will be executed automatically and initialization of data member takes place.
* It is called constructor because it constructs the value of data member of the class.
* A constructor is a member function that is automatically called when a class object is created.
* A constructor is a special member as it has same name of the class.
* when we declare object in the main function, memory will be allocated and constructors will be called by the compiler automatically.

Characteristics of constructors / Syntax Rules of constructor

1. A constructor names must be same as classname in which it is declared
2. It does not have any return value
3. It is generally declared as public member function.
4. It can have default argument
5. It cannot be declared constant, static, variable virtual.
6. It cannot be inherited.

22316

(7) It is automatically called when an object is created.

Syntax for Constructor

① Inside the class

```
class class_name
{
public :
classname(parameters)  //header of constructor (parameter are optional)
{
//body of constructor,
};
};
```

② Outside the class

```
class class_name
{
public :
Class name (parameter.) //constructor
};
class_name :: class name (parameter)
{
//body of Constructor
}
```

Parameters are optional

many constructor is overloaded not overload.

14/09/2023

Example of Constructor:

```
class time
{
  int hour;
  int min;
  int sec;
  public:
    time() // Constructor declared.
};
  time :: time() // definition of constructor.
  {
  hour = min = sec = 0; //Initialize value
  }
```

Here class time is declared with ~~type~~ time constructor. If we create object of the class that is time t when t (object) is created it invoke constructor. that means it initalize data members (hour, min, sec) to 0

Types of constructor (2 marks)
- Default Constructor / Empty Constructor
- Parameterized Constructor
- Copy constructor
- Dynamic Constructor
- Overloaded or multiple constructor

22316

Page No.
Date

Default Constructor/Empty Constructor

* The constructor which does not accept any arguement is called default constructor
* Default Constructor is also called as Empty Constructor because it has no arugement.
* A Default constructor is used to initalize all the objects of the class with same values.
* A Default constsicctor can be called directly when an object of an class is created.
* If no constructor are compiler will create default Constrector by itself.

casting

* Syntax

```
class class name
{
 public :
  class_name()
};
class_name :: class name ()
{
 //body of constructor.
}
```

Q) What is parameterized constructor? (4 marks)

⇒ Sometime it is essential to initialize the various data elements of different object with different values when they are created.
This is achieved by passing arguments/parameters to the constructor functions when objects are created. The constructor which accept any number of formal parameter & thus are used to initialize the object are called parameterized constructors. Parameterized constructor is defined as a constructor which accepts one or more arugements.
At the time of declaration of objects initialize the data members of objects with these arugements.

* Syntax.

```
class classname
{
  .
  .
public:
classname (para 1, para 2, para n)
{
classname :: classname (p1, p2, pn)
}
};
```

22316

```
#include <stdio.h> <iostream>
using namespace std;
class student
{
 int roll_no;
 student(int r);
};
student:: student(int r)
{
 roll_no=r;
}
int main()
{
student(3);
}
```

casting

many constructor
Overloaded

WAP to declare a class student with member as a roll_no, name and department. Declare parameterized constructor with default value for department as 'CO' to initialize member of object. Initialize and display data for two student

```
#include<iostream>
#include<string>
using namespace std;
class student
{
int roll_no;
string name, department;
public:
student(int r, string n, string d="CO")
  {
   roll_no=r;
   strcpy(name,n);
   strcpy(department,d);
  }
void display()
{
cout<<"Roll no: "<<roll_no;
cout<<"Name: "<<name;
cout<<"Dept. name "<<department;
}
};
void main()
{
student s1(112,"abc");
student s2(113,"xyz");
s1.display();
s2.display();
}
```

default argument concept

22316

Output

Roll no: 112
Name: abc
Dept. name: CO
Roll no: 113
Name: xyz
Dept. name: CO

* Write a C++ program to declare a class addition with data members 'x' and 'y' with constructor, calculate addition and display it using function display

casting

```
#include <iostream>
using namespace std;
class addition
{
int x, y;
public:
addition(int x1, int y1)
  {  x=x1;
     y=y1;
  }
void display()
{
cout<<" Addition of two number "<<x+y;
}
};
int main()
{
addition a(3,4)
a.display();
}
```

Output

Addition of two number is 7

* WAP declare class student with data members rollno and name declare a constructor to initialize data members of class.

```
#include<iostream>
using namespace std;
class student
{
 string name;
 int roll_no;
 student ();
 void display ()
 {
  cout<<"student name : "<<name;
  cout<<"Roll No: "<<roll_no;
 }
};
```

```
student::student ()
{
 name="abc";
 roll_no=12;
}
```

```
int main
{
 student x;
 x.display ();
}
```

```
Student name = abc
Roll_no = 12
```

22316

Explain overloading with suitable example.

* C++ allows defining multiple constructors with different number, datatype or order of parameters of a single class using constructor overloading.
* The number of constructor can be defined for same class with different argument list is reffered as overloaded or multiple constructor.
* C++ constructor overloading enables multiple constructors to initalize different objects of a class.
* This objects can be initalize with same values or different values
* In C++, class can have default constructor, parameterised constructor etc,
* The class has multiple constructor is known to be overloading or multiple constructor.
* Syntax

```
Class class_name
{
  public:
  classname()
  {
  //body of default constructor
  }
  class name (para 1, para 2)
  {
  //body of parameterised constructor
  }
}
```

Example,

```
#include <iostream>
using namespace std;
class Abc
{
 int a,b;
 public:
  Abc()
 {
  a=150;
  b=100;
 }
 abc (int x, int y)
 {
   a=x;
   b=y;
  }
  void display ()
  {
   cout <<"values "<<a<<b;<<b;
  }
};
int main ()
{
 Abc a(10,20);
 abc b;
 a.display ();
 b.display ();
 return 0;
}
```

Output

values 10 20
values 150100

22316

Page No.
Date

Constructor with default arugements

① No. of times function uses same arugements for some situations for example, while calculating the total prize of the items Same discount rate many applies to all the particular products.

② Therefore, instead of suppling such arage ments all the time C++ allow us to define an arugement whose value will be automatica lly used by the ~~compilies~~ complier. casting

③ If it is not provided during the function call this arugement is called default arugement.

④ whenever call is made to the function without specifing an arugement the program will automatically assign values to the parameter which is default

⑤ Default arugements are useful when user want same value for arguement.

example

21/9/23

26/09/2023

Page No.
Date

Copy Constructor

* Copy Constructor takes reference to an object of the same the class as itself as an argument.
* A constructor that initialize an object using values of another objects pass to it as parameter is called Copy constructor.
* In c++, a new object of a class can also be initialise with existing object of the same class. For this, compiler of C++ calls Copy constructor.
* Copy constructor creates the copy of passed object
* A copy constructor is defined as constructor which accepts an already existing object through reference to copy data member values.
* Syntax: Copy constructs as follows

```
class class_name
{
public :
class_name (class_name &object_name)
{
// body of constructor
};
```

22316

Example

```
#include <iostream>
using namespace std;
class abc
{
  int x=0, y=12;
public:
  Abc();                  // default constructor
  Abc(Abc &);             // copy constructor.
void display()
{
  cout<<x;
  cout<<y;
}
};
Abc::Abc(Abc &a)
{
 x=a.x;
 y=a.y;
}
int main()
{
Abc a;
Abc b(a); //invoking copy constructor
a.display();
b.display();

}
```

Output

10 20

10 26

Destructor

* A Destructor is a special member function that is automatically called when object is destroyed.
* Destructor are member functions with same name as the class, preceed by tilde character (~)
* For example, the destructor for the circle class could be named ~circle
* In a same way that constructors set things up, when an object is created destructor performs shut down procedure when the object goes out of existence.
* The primary uses of destructor function is to release space on the heap.
* whenever a particular goes out of the scope destructors will be called automatically and it takes out the allocated memory.
* Destructors are invoked by the compiler implicitly when the termination of the program takes place but it is always better to invoked destructor explicitly by writing it into program code.

Syntax

```
class class_name
{
  public:
  class_name ( ) { }
   ~ classname ( )
     {
     // body of destructor
     }
};
```

22316

example

```
class Sample
{
// data variable
// member functions
public:
Sample ()      // constructor
~ Sample ()    // destructor
};
```

Parameter	Constructor	Destructor
Purpose	It allocates the memory to an object.	It deallocates the memory of an object.

Syntax	class : class name
	{
	public :
	classname (parameter)
	{
	// body of constructor
	}
	};
		class class_name
		{
		public :
		class_name ; { }
		~ class name ()
		{
		// body of destructor
		}
		};

Arrgement	Constructor accepts arrgement example parameterized constructor.	Destructor does not accept any arrgement.
Calling	Constructor is automatically when an new object is created.	Destructor is called by compiler implicitly when the termination of program takes places.

Name	Constructor has same name as class name	Destructor has same name as class but with tilde (~).
Use	Constructor is used for initializing value to the data members of the class.	Destructor is used when object is destroyed.
Number	There are can be multiple constructor in class.	There is always single destructor in class
Overloading	Constructors can be overloaded.	Destructors cannot be overloaded.
Return type	~~Return type = void, int~~ Constructor does not have return type	It does not uses return type.
types	In c++ we can use different types of constructor → Copy → Parameterized → Default → Dynamic	Destructor does not has types.

estructor has same
ame as class but with
ilde (~).

ructor is used when
t is destroyed.

e is always single
uctor in class

tors cannot
erloaded.

not uses
type.

n does not
es.

* Syntax Rules for destructors (>m/Um)

(1) Destructor name must be same as class name. But prefix by tilde (~)
(2) It does not has any return type.
(3) It is declared as public member fun^n.
(4) It takes no argument and therefore cannot be overloaded.
(5) It cannot be declared static, constant, variable.
(6) It is automatically called when object is destroyed.
(7) It specifies how an object is deleted.

* Uses of destructor

(1) When an object is destroyed then the destructor is called.
(2) The primary use of destructor function is to realised space on the heap.
(3) Whenever a particular object goes out of scope destructor will be called automatically and it takes out automatically the allocated memory.
(4) It also do other clean-up for the class object.

casting