2. Functions and Constructure

PAGE NO.:

Inline Functions :

If a function is to be define outside the class but still be created as a Internally defined function then such functions are clalled as Inline Function.

Syntax :

```
inline Return_type Class_name :: Function name
{                                 (argument list)
  Statement;
}
```

Q. write a program to find out area of circle using oop such that the class circle must have pre-Inline functions namely.
read() → To accept the radius
Compute() → To calculate the area.
display() → To display the result.

→
```
#include <iostream.h>
#include <conio.h>
Class circle
{
 Public :
 int r;
 float a;
 Void read();
 Void compute();
  Void display();
};
```

```
inline Void circle :: read()
{
 Cout<< "Enter r";
 Cin >>r;
}
inline Void circle :: Compute()
{
   a = 3.14 * r * r;
}
inline void circle :: display()
{
 Cout<< "area of circle" << a;
}
Void main()
 {
 circle c;
 c. read();
 c. Compute();
 c. display();
 }
```

Q2. Write a program to find out sum of following series using Inline function.

$2^2 + 4^2 + 6^2 + 8^2 + \dots + n^2$

→

```
#include <iostream.h>
#include <conio.h>
Class Series {
Public :
 int n, i, sum;
```

```
Void read();
Void compute();
Void display();
};

Inline Void Series :: read()
{
 Cout << "Enter the n";
 Cin >> n;
}

Inline Void Series :: compute()
{
 Sum = 0;
 for (i=2; i<=n; i++)
 {
  Sum = Sum + (i*i)
 }
}

Inline Void series :: display()
{
  Cout<<"Sum of Series"<< sum;
}

Void main()
{
  Series s;
  s.read();
  s.compute();
  s.display();
}
```

PAGE NO.:

Constructor:

They are a special member function used for initializing the data elements of the object

⇒ Properties or charecterstics of constructor.

1] The name of constructor must always be same as that of class.

2] It has to be defined only in the public visibility of the class

3] It should not have any return type not even void.

4] The constructor is executed the movement an object of a class is created.

5] A constructor cannot be inherited but a derived class can call the constructor of the base class.

6] A constructor can not be static or virtual

7] Any number of constructor can be define in a class and such a set of constructor implements a concept called as constructor overloading.

PAGE NO.:

Types of Constructor

1] default Constructor
2] Parameterised Constructor
3] Copy Constructor.

1] Default Constructor:
If a constructor does not have any Parameter list then it is called as default constructor.

Q. Write a program to find Area of Circle using oop the value of the radius must be accepted from the user in the constructor. and the Class Circle must have two inline functions compute(), → Calculating area.
display() → display area.

→
```
#include <iostream.h>
#include <conio.h>
Class circle
{
Public:
 int r;
 Float a;
 circle()
{
Cout<< " Enter r";
Cin>>r;
}
 Void compute();
```

```
Void display();
};

inline Void circle :: compute()
{
  a = 3.14 * r * r;
}

inline Void circle :: display()
{
  Cout<< " Area of circle:" << a;
}

Void main()
{
  circle c;
  c. compute();
  c. display();
}
```

2] Parameterised Constructor:
if a constructor does have a parameter list that is which accept one or more parameters then it is called as Parameterised Constructor.

Syntax:

Class_name object_name (argument list);

Q. Write a program to find out area of circle using oop the value of the radius must be accepted from the user in the main program and passed to the Parameterised Constructor. and the class circle must have two inline function namely compute () → to calculate area display() → to displaying the reason.

→
```
#include <iostream.h>
#include <conio.h>
Class circle
{
Public:
 int a, r;
 circle ( float x)
 {
 r = x;
 }
 Void compute();
 Void display();
};
```

```
inline Void circle::compute()
{
a=3.14*r*r;
}
inline Void circle:: display()
{
 cout<<" Area :"<<a;
}
Void main()
{
cout<<"Enter r";
cin>>r;
circle1 c(r);
c.compute();
c.display();
}
```

3] Copy constructor :
If a constructor has an object of the same class as a parameter in the argument list that is called as copy constructor.

The copy constructor uses an object of the same class to initialize another object of a class.

Q1 Write a program to find area of circle using oop the value of the radius must be accepted from the user in the main program and passed to the copy constructor and the class circle must have two inline functions namely
Compute → To calculate area
display → For displaying the result

→
```
#include <iostream.h>
#include <conio.h>
Class circle
{
Public :
int a, r ;
circle (float x)
{
 r = x;
}
circle (circle & c)
{
   r = c.r;
}
```

```
Void compute();
Void display();
};

inline Void circle :: compute()
{
a = 3.14 * r * r;
}
inline Void circle :: display()
{
cout << "area = " << a;
}

Void main()
{
cout << "Enter radius";
cin >> r;
circle c(r);
c.compute();
c.display();
circle c1(r1);
c1.compute();
c1.display();
}
```

* Destructor:

A destructor is used to destroy the memory Allocated by the constructor

Carecteristics of destructor.

- It is defined only in the public visiblity of class
- The name of the destructor must be same as that of class prefix with a tilde sign (~)
- It cannot return or accept any value it is automaticly called when the compiler exits from the main program.
- The destructor can only destroy the memory location allocated by the constructor and the constructor can allocate memory only using the new operator.

- To destroy this memory allocated by the new operator we require delete operator in this case the delete operator must be in the destructor.

```
#include <iostream.h>
#include <conio.h>
class test
{
 public:
 int *p;
 test()
 {
  p = new int;
 }
 void read()
 {
 cout << "Enter Value";
 cin >> *p;
 }
 Void display()
 {
  cout << " Value : " << *p;
 }
  ~ test()
 {
   delete p;
  cout << "Destroyed";
 }
 };
```

```
Void main()
{
test t;
t.read();
t.display();
}
```

Difference between Constructor & Destructor

Constructor	Destructor
(I) A Constructor is a special member function whose task is to insalize the object of its class	(I) A Destructor is a special member function whose task is to destroy the object of that have been created by the Constructor.
(II) Constructor are classified in various type such as ① Default Constructor ② parameterised ③ Copy Constructor ④ overloded Constructor or multiple Constructor.	(II) Destructors are not classified in any type
(III) It is invoke automaticly when the objects are created	(III) It is invoke implicitaly by the compiler upon exit of Program.

(IV) Constructor accept parameters also it can have default value for its parameter	(IV) Destructor never accept parameters
(V) A class can have more than one constructor	(V) A class can have at the most one destructor
(VI) It construct the values of data member of a class	(VI) It does not construct the values for the data member of the class

(VII) Syntax:
Classname()
{
};
	(VII) syntax:
	~Classname()
	{
	};

- Static Data member and static function member.

Static members are common for the entire class and not for each object of the class we can have static Data members as well as static function members

Q. write a program to define a class having Data members principal, duration & rate of interest declared rate of interest as static member variable. Calculate the simple interest & Display it.

→

```
#include <stdio.h>
#include <conio.h>
Class interest
{
Public :
int p, d;
float SI;
Static float r;
Void accept()
{
Cout<<"Enter Principal duration";
cin>>p>>d;
}
Void calculate()
{
SI = pxdxr / 100;
}
```

```
Void display()
{
cout<< " simple interest :"<<SI;
}
};

Float interest :: r=15;
Void main()
{
interest i;
i.accept();
i.calculate();
i.display();
}
```

* Compare static and non-static datamembers.

Static	Non-Static
(i) It is declare using static keyword.	(i) It is declared without using static keyword.
(ii) Static data member is automaticly initialize with zero	(ii) Non-static data member can be inilialize with zero or non-zero
(iii) All object of a class share same copy of static datamembers	(iii) Each object of a class get its own copy of non-static data members.

(iii) It can be access static member function as well as non static member function of the class

(iv) It can be access only the non-static member function of the class.

(iv) Syntax:
Static datatype
variable name;

(v) Syntax:
datatype Variable_name

⇒ Array of object
When multiple objects of a class are to be made we can use array of object

Syntax:
class_name object_name [size];

Q. write a program to declare a class staff having data members as name, dept, accept this data for 10 staff & display name and department of the staff.

→
```
#include <iostream.h>
#include <conio.h>
class staff
{
public:
char name[20], dept[20];
void accept()
{
```

```
cout << " Enter name & dept";
cin >> name >> dept ;
}
Void display ()
{
cout << name << "\t" << dept ;
}
}
Void main()
{
Staff S[10];
int i;
for(i=0; i<=9;i++)
{
S[i] .accept();
}
for(i=0; i<=9; i++)
{
S[i]. display();
}
}
```

Q Write a program to declare a class account having datamember as account_no & balance accept and display data for 5 object using array of object.

→
```
#include <iostream.h>
#include <conio.h>
Class Account
{
```

```
Public:
int account_no;
float balance;

Void accept()
{
cout << "Enter account no";
cin >> account_no;
cout << " Enter balance";
cin >> balance;
}

Void display()
{
cout << "Account No:" << account_no;
cout << " Balance:" << balance;
}
};
Void main()
{
  Account a[5];
  int i;
  for (i=0; i<=4; i++)
  {
  a[i].accept();
  }
  for (i=0; i<=4; i++)
  {
   a[i].display();
  }
}
```