

object-oriented programming (OOP)

KIAN _ ACADEMY

Passing and Returning Objects in C++:

In C++ we can pass class's objects as arguments and also return them from a function the same way we pass and return other variables. No special keyword or header file is required to do so

Example on pass by value

```
Float add(float num1, float num2)
{
    return num1 + num2;
}

string add(string a, string b)
{
    return a + " " + b;
}
```

Example on Passing an Object as argument

```
#include <bits/stdc++.h>
using namespace std;
class Example {
public:
    int a;
    Example add(Example Ea, Example Eb)
    {
        Example Ec;
        Ec.a = Ea.a + Eb.a;
        return Ec;
    }
};
```

```
int main()
{
    Example E1, E2, E3;
    E1.a = 50;
    E2.a = 100;
    E3.a = 0;
    cout << E1.a << endl;
    cout << E2.a << endl;
    cout << E3.a << endl;
    E3 = E3.add(E1, E2);
```

cout << "-----" << endl;
cout << E1.a << endl;
cout << E2.a << endl;
cout << E3.a << endl;
return 0;

Output:

50
100
0

50
100
150

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
class Rectangle
{
private:
    int width,height;
public:
    Rectangle();
    Rectangle(int w, int h );
    ~Rectangle();
```

```
int area()
{
    return(width * height);
}
Rectangle addRectangle(Rectangle r);
#endif // RECTANGLE_H

#include "Rectangle.h"
Rectangle::Rectangle(int w, int h )
{
    width=w;
    height=h;
}
```

```
Rectangle::Rectangle()
{
}
Rectangle::~Rectangle()
{
}
Rectangle
Rectangle::addRectangle(Rectangle r)
{
    Rectangle result;
    result.height=height+r.height;
    result.width=width+r.width;
    return result;
}
```

```
#include<iostream>
#include<Rectangle.h>
using namespace std;
int main()
{
    Rectangle r1(1,2);
    Rectangle r2(3,4);
    Rectangle r3=r1.addRectangle(r2);
    cout<<r1.area()<<endl;
    cout<<r2.area()<<endl;
    cout<<r3.area()<<endl;
}
```

Static Class Members:

- ▶ We can define class members static using **static** keyword
- ▶ All static data is initialized to zero when the first object is created
- ▶ - if no other initialization is present.
- ▶ -We can't put it in the class definition but it can be initialized outside the class .
- ▶ As done in the following example by redeclaring the static variable, using the scope resolution operator :: to identify which class it belongs to.
- ▶ **why we using static members?**
 - 1 → to share all objects of the class
 - 2 → to reduce space in RAM
 - 3 → as counter to store the number of objects in the class.
- ▶ **Note:**

A static member is shared by all objects of the class.

Example :-

```
#include <iostream>
using namespace std;
class Box
{
private:
    double length;
    double breadth;
    double height;
public:
    static int objectCount;
    Box(double l = 2.0,
        double b = 2.0,
        double h = 2.0)
    {
        length = l;
        breadth = b;
        height = h;
        objectCount++;
    }
    double Volume()
    {
        return length * breadth * height;
    }
};
int Box::objectCount = 0;
```

```
int main(void)
{
    Box Box1(3.3, 1.2, 1.5);
    Box Box2(8.5, 6.0, 2.0);
    cout << "Total objects: " <<
    Box::objectCount << endl;
    return 0;
```

output:

Total objects: 2

Static Function Members

- ▶ By declaring a function member as static
 - you make it independent of any particular object of the class.
 - A static member function can be called even if no objects of the class exist.
 - the static functions are accessed using the class name or the object and the scope resolution operator (::)
 - A static member function can only access static data member
 - other static member functions and any other functions from outside the class.
 - Static member functions have a class scope
 - they do not have access to the this pointer of the class.
 - You could use a static member function to determine whether some objects of the class have been created or not.

Example :-

```
#include <iostream>
using namespace std;
class Box
{
private:
    double length;
    double breadth;
    double height;
public:
    static int objectCount;
    Box(double l = 2.0, double b =
2.0, double h = 2.0)
```

```
{
    length = l;
    breadth = b;
    height = h;
    objectCount++;
}
double Volume()
{
    return length * breadth * height;
}
static int getCount()
{
    return objectCount;
}
```

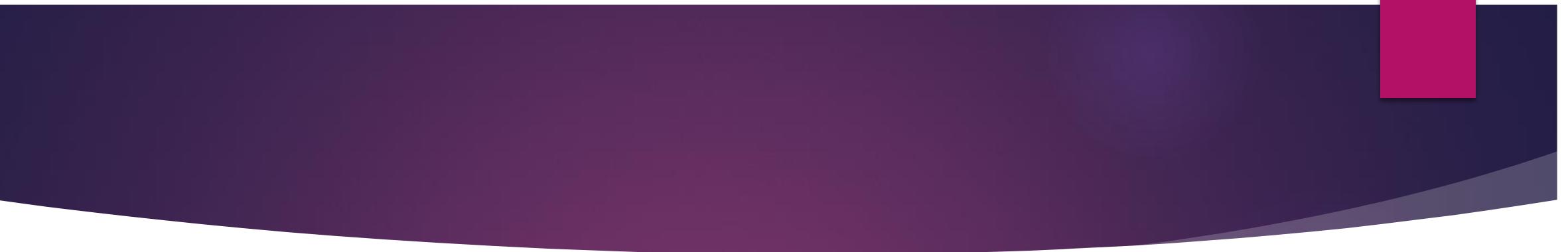
```
int Box::objectCount = 0;
int main(void)
{
    cout << "Initial Stage Count: " << Box::getCount()
<< endl;
    Box Box1(3.3, 1.2, 1.5);
    Box Box2(8.5, 6.0, 2.0);
    cout << "Final Stage Count: " << Box::getCount()
<< endl;
    return 0;
}
```

```
*****
output:
Initial Stage Count: 0
Final Stage Count: 2
```

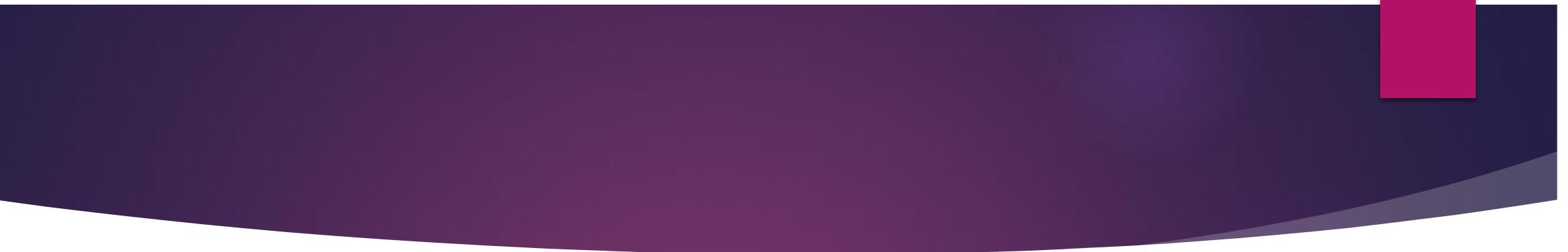
Exam questions:

- ▶ (1) Passing object to a function -----

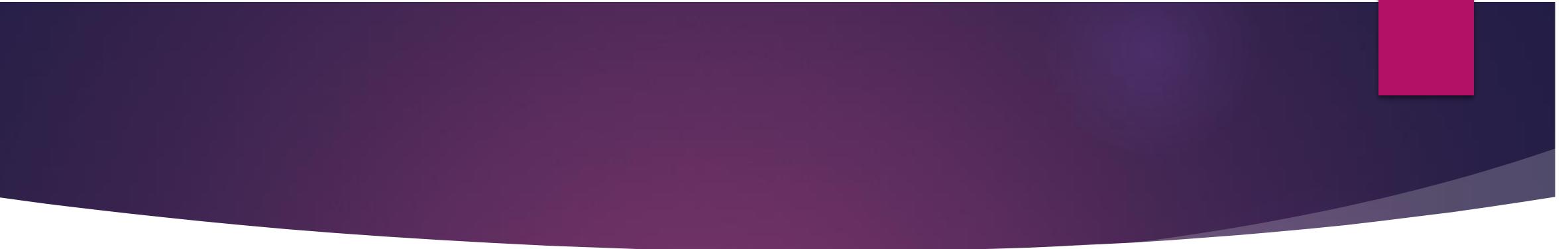
- ▶ A) Can be done only in one way
- ▶ B) Can be done in more than one ways
- ▶ C) Is not possible
- ▶ D) Is not possible in OOP



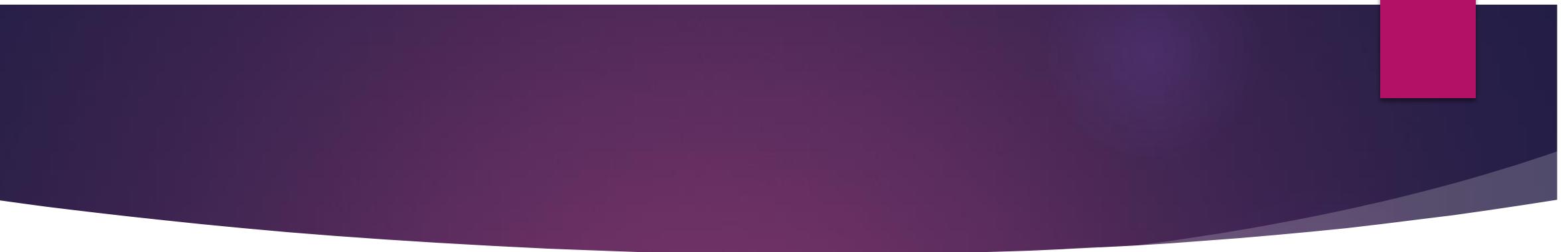
- ▶ (2) The object -----
 - ▶ A) Can be passed by reference
 - ▶ B) Can be passed by value
 - ▶ C) Can be passed by reference or value
 - ▶ D) Can be passed with reference



- ▶ (3) Which symbol should be used to pass the object by reference in C++?
 - ▶ A) &
 - ▶ B) @
 - ▶ C) \$
 - ▶ D) \$ or &

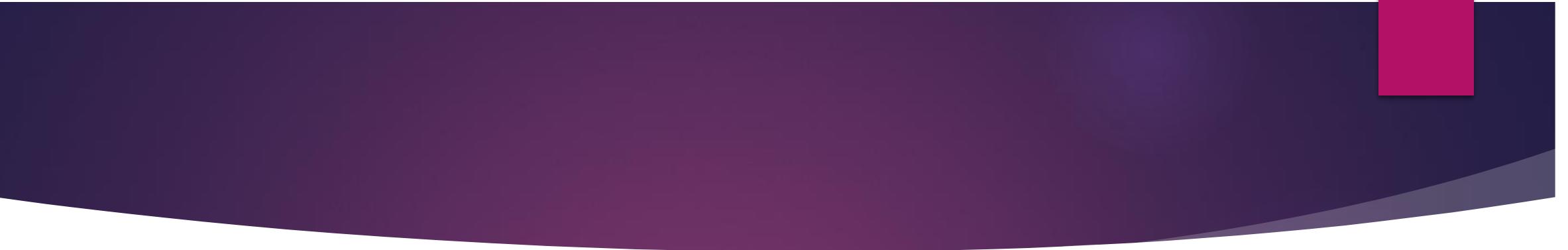


- ▶ (4) If object is passed by value -----
 - ▶ A) Copy constructor is used to copy the values into another object in the function
 - ▶ B) Copy constructor is used to copy the values into temporary object
 - ▶ C) Reference to the object is used to access the values of the object
 - ▶ D) Reference to the object is used to created new object in its place

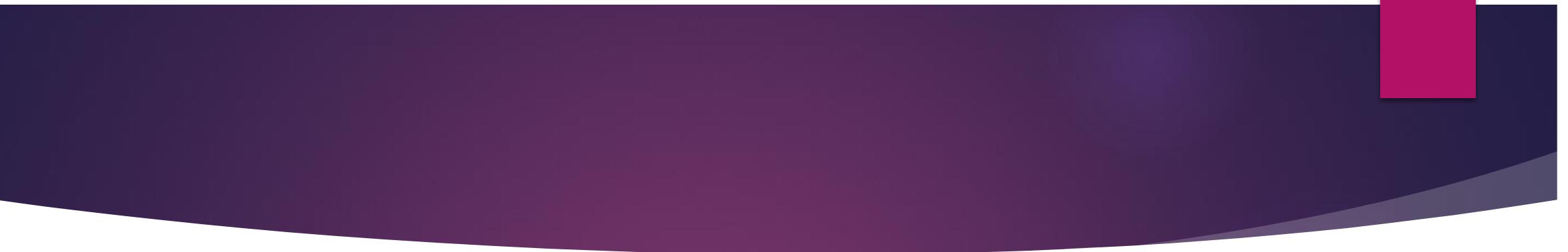


- ▶ (5) Pass by reference of an object to a function -----

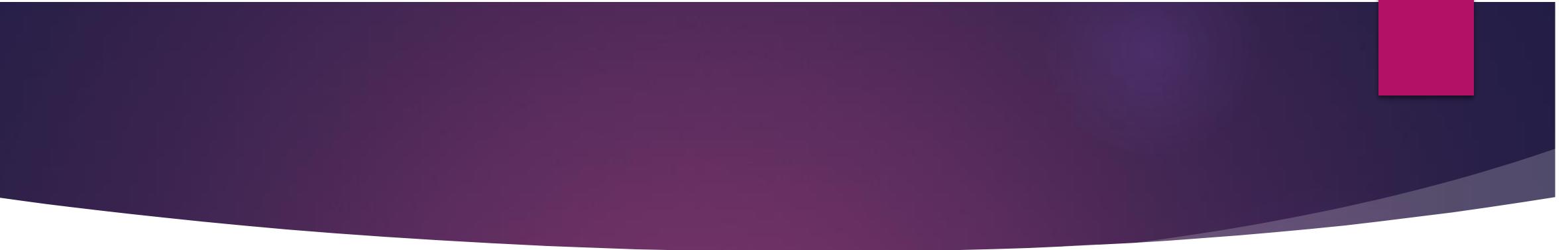
- ▶ A) Affects the object in called function only
- ▶ B) Affects the object in prototype only
- ▶ C) Affects the object in caller function
- ▶ D) Affects the object only if mentioned with & symbol with every call



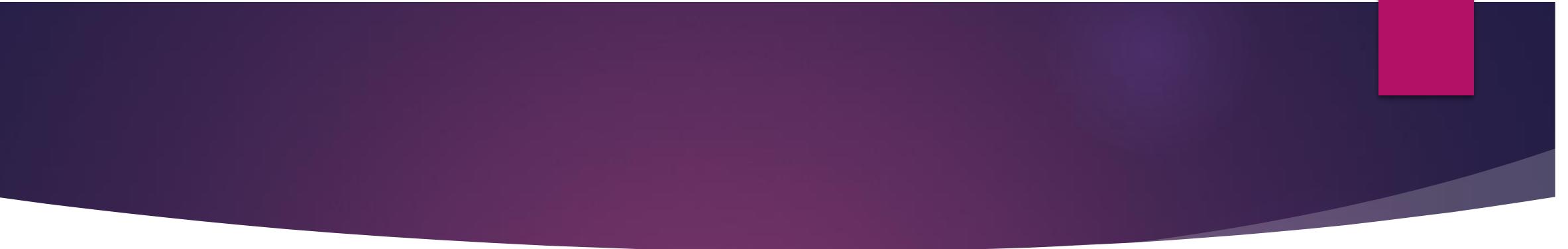
- ▶ (6) Copy constructor definition requires -----
 - ▶ A) Object to be passed by value
 - ▶ B) Object not to be passed to it
 - ▶ C) Object to be passed by reference
 - ▶ D) Object to be passed with each data member value



- ▶ (7) What is the type of object that should be specified in the argument list?
 - ▶ A) Function name
 - ▶ B) Object name itself
 - ▶ C) Caller function name
 - ▶ D) Class name of object



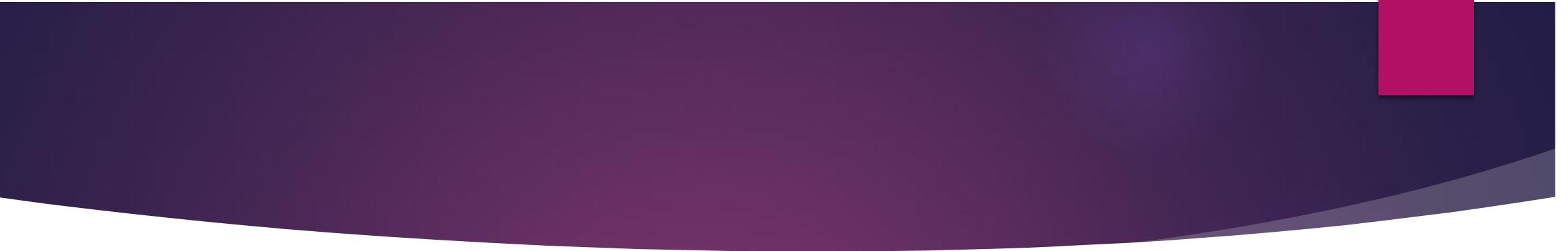
- ▶ (8) If an object is passed by value, -----
 - ▶ A) Temporary object is used in the function
 - ▶ B) Local object in the function is used
 - ▶ C) Only the data member values are used
 - ▶ D) The values are accessible from the original object



- ▶ (9) Which among the following is correct definition for static member functions?
 - ▶ A) Functions created to allocate constant values to each object
 - ▶ B) Functions made to maintain single copy of member functions for all objects
 - ▶ C) Functions created to define the static members
 - ▶ D) Functions made to manipulate static programs

► (10) The static member functions -----

- A) Have access to all the members of a class
- B) Have access to only constant members of a class
- C) Have access to only the static members of a class
- D) Have direct access to all other class members also



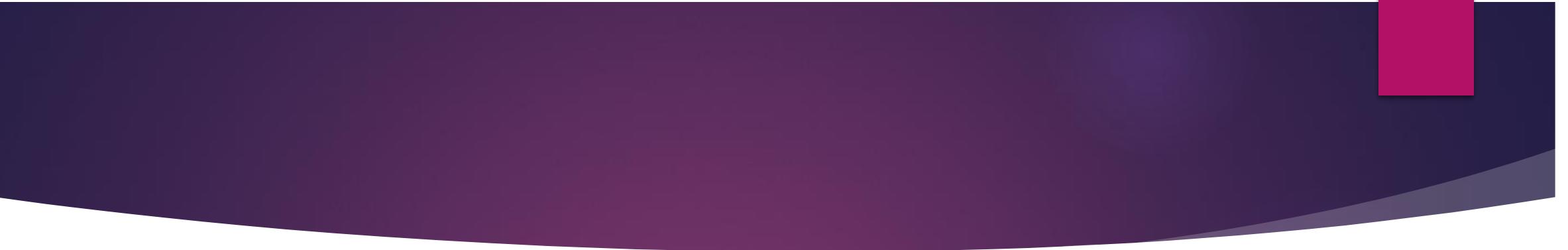
► (11) The static member functions -----

- A) Can be called using class name
- B) Can be called using program name
- C) Can be called directly
- D) Can't be called outside the function

- ▶ (12) Which is correct syntax to access the static member functions with class name?
 - ▶ A) className . functionName;
 - ▶ B) className -> functionName;
 - ▶ C) className : functionName;
 - ▶ D) className :: functionName;

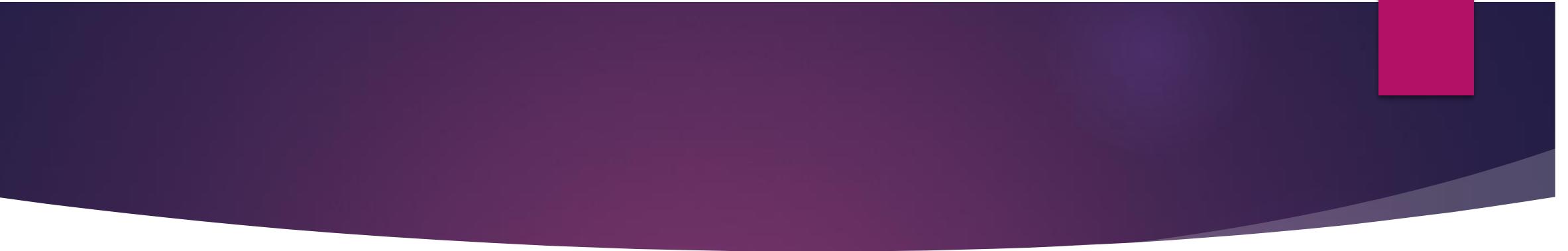
► (13) Which among the following is not applicable for the static member functions?

- A) Variable pointers
- B) void pointers
- C) this pointer
- D) Function pointers

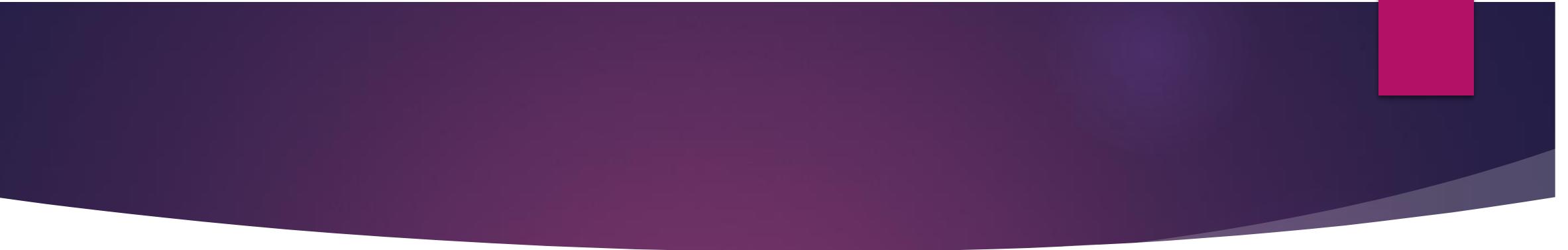


- ▶ (14) Which among the following is true?
 - ▶ A) Static member functions can't be virtual
 - ▶ B) Static member functions can be virtual
 - ▶ C) Static member functions can be declared virtual if it is pure virtual class
 - ▶ D) Static member functions can be used as virtual in Java

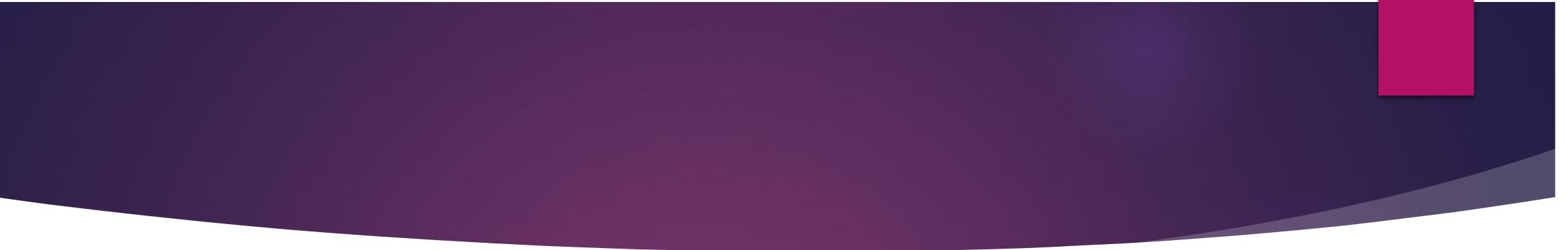
- ▶ (15) The static members are----
 - ▶ A) Created with each new object
 - ▶ B) Created twice in a program
 - ▶ C) Created as many times a class is used
 - ▶ D) Created and initialized only once



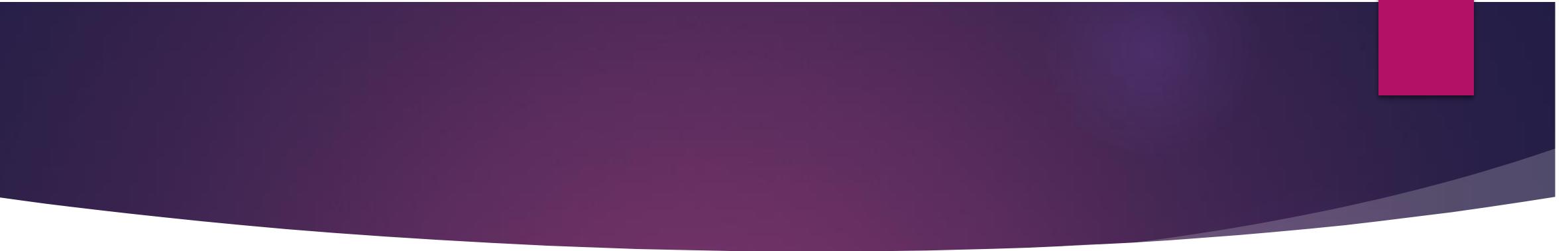
- ▶ (16) The static member functions ---
 - ▶ A) Can't be declared const
 - ▶ B) Can't be declared volatile
 - ▶ C) Can't be declared const or volatile
 - ▶ D) Can't be declared const, volatile or const volatile



- ▶ (17) Which keyword should be used to declare the static member functions?
 - ▶ A) static
 - ▶ B) stat
 - ▶ C) const
 - ▶ D) common



- ▶ (18) The keyword static is used ---
 - ▶ A) With declaration inside class and with definition outside the class
 - ▶ B) With declaration inside class and not with definition outside the class
 - ▶ C) With declaration and definition wherever done
 - ▶ D) With each call to the member function



► (19) The static data member-----

- A) Can be mutable
- B) Can't be mutable
- C) Can't be integer
- D) Can't be characters

Answer the questions :-

- | | | |
|---------|----------|----------|
| ► 1 - b | ► 7 - d | ► 13 - c |
| ► 2 - c | ► 8 - b | ► 14 - a |
| ► 3 - a | ► 9 - b | ► 15 - d |
| ► 4 - a | ► 10 - c | ► 16 - d |
| ► 5 - c | ► 11 - a | ► 17 - a |
| ► 6 - c | ► 12 - d | ► 18 - b |
| | | ► 19 - b |