01-07-2024

CONSTRUCTORS ROLE IN INHERITENCE =>

when sub-class object is created not only sub-class constructor but also super-class constructor is called.

1st super class constructor body is executed then after that sub-class constructor body is executed.

EX:-

2

-> using namespace std; .CPP

```
#include <iostream>
class abc
{ public: abc()
          {
            cout << "abc constructor called ";
          }
          abc(int x)
          { cout << "abc int constructor called \n";
          }
};
class xyz: public abc
{
  public: xyz(int x, int y): abc(x)
          {
            cout << "xyz constructor called \n";
          }
};
```

```
main()
{
    xyz p(1,2);
}
```

O/P:-
abc int constructor called
xyz constructor called

=> using namespace std;
#include <iostream>
class abc
{
    public: abc()
        {
            cout<<"abc constructor called \n";
        }
};
class xyz
{
    public: xyz()
        {
            cout<<"xyz constructor called \n";
        }
};
class mno: public abc, public xyz
{
    public: mno()
        {
            cout<<"mno constructor called \n";
        }
};
main()
{
    mno p;
}

O/P:-
mno constructor called

REFERENCE :-

```
-> using namespace std;
   # include <iostream>
   main ()
   {
      int a = 5, &b = a;
      b = 10;
      cout << a;
   }
```

o/p :- 10

-> a reference b
[OR]
a alias b

call by value
call by reference

CALL BY REFERENCE =>

```
-> using namespace std;
   # include <iostream>
   void swap (int &a, int &b)
   {
      int c;
      c = a; a = b; b = c;
   }
   main ()
   {
      int a = 5, b = 10;
      swap (a, b);
      cout << "a = " << a << " b = " << b << "\n";
   }
```

o/p :-
~~5, 10~~
10, 5

.cpp

=

[OR]

```
main ()
{
   int a, b;
   printf ("Enter a : ");
   scanf ("%d", &a);
   printf ("Enter b : ");
   scanf ("%d", &b);
   swap (a, b);
   cout << "a = " << a <<
   " ; b = " << b << "\n";
}
```

o/p :-
Enter a : 5
Enter b : 10
a = 10 ; b = 5

TEMPLATE [OR] GENERICS :-

Templating is a method of writing of 1 function [OR] 1 class for family of similar functions [OR] similar class in generic manner.

FUNCTION TEMPLATE :-

```
-> Using namespace std;
# include <iostream>
template < typename t >
void swap ( t *a, t *b)
{
  t c;
  c = *a ; *a = *b; *b = c;
}
main ()
{
   int a=5, b=10;
   float x=2.5f, y=5.2f;
   char m='a', n='b';
   swap (&a, &b);
   swap (&x, &y);
   swap (&m, &n);
   cout << "a=" << a << "b=" << b << "\n";
   cout << "a=" << x << "b=" << y << "\n";
   cout << "a=" << m << "b=" << n << "\n";
}
```

O/P :-

```
Enter a : 5
Enter b : 10
Enter x : 2.5
Enter y : 5.2
a = 10  b = 5
x = 5.2  y = 2.5
   b       a
```

CLASS TEMPLATE :-

-> using namespace std;

```
#include <iostream>
template < typename t >
class abc
{
    t a, b;
    public : void get ()
        {
            cout << "enter a, b";
            cin >> a >> b;
        }
        void put ()
        {
            cout << a << b;
        }
};
main ()
{
    abc < int > p;
    abc < float > q;
    p.get (); p.put (); q.get (); q.put ();
}
```

.cpp

O/P :-

```
Enter a; b : 5
  10
5 10
Enter a, b : 2.5
3.5
2.5  3.5
```