$\boxed{\text{Function Overloading in C++}}$

O/L^g = overloading // Short terms

$$\begin{tikzcd}
 & \text{Polymorphism} \arrow[dl] \arrow[dr] & \\
\text{fun } O/L^g & & \text{Operator } O/L^g
\end{tikzcd}$$

- Polymorphism is a general term
  Poly means multiple
  morphism " shape.

- PM means multiple use of same entity

- In C++, there r two ways to implement PM - fun O/L^g
  - Operator O/L^g.

<u>Q</u> What is fun O/L^g? Why is it done?

<u>A</u> It means, there are several fun with same name.
FOL is only allowed in C++
but not in C

• ~~It is done, so that mul fun can have same name~~

• It is done to allow having same name for different fun. So, that user doesn't need to have different name for similar fun.

eg Suppose, there r ~~df~~ different fun for adding int, char, floats

$$\begin{tikzcd} \boxed{\text{Sum}} \arrow[d] & \boxed{\text{Sum}} \arrow[d] & \boxed{\text{Sum}} \arrow[d] \\ \text{add int} & \text{add float} & \text{add char.} \end{tikzcd}$$

• User can use same name for 3 different fun.

Q What is the rule for FOL?

A • Different fun can have same name

*** • But parameters should be different

- No. of parameter ~~should~~ $\overset{\text{can}}{}$ be diff
- type of " can " "

eg void sum(int x, int y); // 2 param of type int
void sum(int x, int y, int z); // 3 param of type int
void sum(char x, int y); // 2 param of types char and int.

↑
These r 3 overloaded fun with different param.

EXAMPLE of FOL - 1

Q Do not use class.
O/L a sum fun for - 2 int
- 3 int
- 1 char, 1 int

Answer
```
#include <iostream>
using namespace std;
void ~~sum(int x, int~~
// define 3 ~~sum~~ sum fun w/o any class.
void sum (int x, int y) // sum fun w/ 2 int.
{
    int add = x+y;
    cout << add;

}
```

```
//Define sum with 3 int
Void sum(int x, int y, int z)
{
    int add = x+y+z;
    Cout<<add;

}

//Define sum with 1 char, 1 int
void sum( char x, int y)
{
    char
    ~~int~~ add = x+y;
    Cout<< add;

}.

int main()
{
    int a=10, b=20, c=30;
    Char ch= 'a';
    sum(a,b); //o/p = 30
    sum(a,b,c); //o/p = 60.
    sum(ch,1); //o/p = b
    //ch is 'a'. 1 will be added
    to ascii value of 'a' which
    will become ascii value of 'b'
    so o/p will be b.

}
```

Example of FOL-2

Q overload a class member fun.

- overload the set fun in student class
  Create 2 set fun
  1) with 3 parameter
  2) with 1 paramter. It will only assign value to rn.

A
```
#include<iostream>
using namespace std;
class student
{
        private:
        int rn, m1, m2;
        public:
        //set fun w/ 3 Param.
        void set (int x, int y, int z);

       //another set fun w/ 1 param
       void set (int x);

       void get ();

};
void student :: set (int x, int y, int z)
{
        rn=x;
        m1=y;
        m2=z;

}
```

```
//Define the second set fun
Void student:: set (int x)
{
        rn = x
        //It will assign values
         to only rn;

}

void student:: get()
{
        cout<<rn<<m1<<m2;

}

int main()
{
        student obj1, obj2;
        obj1. set(5, 50,50); //rn=5 is
        obj1. get();                assigned

        obj2. set (2, 30,30);
        obj2-get();

        obj1. set(1); //change rn of
                                    obj1.
        //assign rn=1 to obj1
        obj1. get();
}

        O/P:-
        5,50,50.
        2,30,30
        1,50,50.
```

Constructor Overloading

Q What is constructor overloading? Why is it done?

A
- Constructor is a type of fun.
- COL means to create multiple constructors in a class.
- It is done to have flexibility while creating objects of a class

Eg If there is only 1 const. in class then obj can be created only by passing values. We can't create obj w/o passing values.

① student obj 1; // There will be an error, bcoz there is a parameterized const. in class.

② Student obj1(1,10,20); // This is correct. Values shud be passed while creating obj.

In this eg., there is only 1 const. so obj can only be created by passing values.
- We have no option of creating obj w/o Passing values.

• Now with the help of COL, we can have flexibility to create obj both ways - with passing values and without passing values.

<u>Q</u> How is ~~to~~ COL done?

• WAP with 2 const. Create a student class

• Create 2 const

1) with parameter
2) w/o Param. i.e a blank const.

• Now, create obj in two ways

1) By passing values

```
student obj1 (1, 10, 20); // Parameterized
                             const called.
```

2) ~~By~~ w/o Passing values

```
student obj 2; // Blank const.
                  is called.
```

<u>Answer</u>

```
#inclu....
Using ....
class student
{
    int rn, m1, m2;
    public:
    void set (int x, int y, int z);
    void get ();
    Student (int x, int y, int z); // Const. w/ Param
    Student (); // No param.
};
```

//In the previous class, two const
are declared :
1) Parameterized Const
2) Non -Parameterized const.

```
void studt :: set(int x, int y, int z)
{
        rn = x;
        m1 = y;
        m2 = z;

}

void student :: get ( )
{
        Cout<<rn<<m1<<m2;

}
~~void~~ //define Parameterized const.
~~void~~ stud :: Studt (int x, int y, int z)
{
        rn = x;
        m1 = y;
        m2 = z;
        Cout<< "Const Called";
}
//Define non-parameterized const.
  student :: Studet ( )
{
        cout<< "Const Called";

}
```

```
int main()
{
    student obj1(1,10,20), obj2;
    // obj 1 will call parameterized const
    // obj 2   "    "   non parameterized
                        const.

    obj1.get();
    obj2.set(2,22,22);
    obj2.get();

}
```

O/P:-

```
1, 10, 20.
2, 22, 22.
```

Summary

$$\begin{tikzcd}
 & \text{Polymorphism} \arrow[dl] \arrow[dr] & \\
\text{fun O/L} \arrow[d] & & \text{Operation O/L.} \\
 & &
\end{tikzcd}$$

Eg 1:- O/L sum fun three times
Eg 2:- O/L set fun in student class 2 times.

Eg 3: Const. overloading - Overload the Const 2 times.