Polymorphism

- The word **polymorphism** means having many forms. Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.

poly + morphism

Can be achieved by

- Early Binding
  - Function Overloading
  - Operator Overloading
- Late Binding
  - Virtual Function

Function Overloading

```
#include <iostream.h>
 class san
{
public:
void print(int i) { cout << "Printing int: " << i << endl;
void print(double f) { cout << "Printing float: " << f << endl; }
void print(char* c) { cout << "Printing character: " << c << endl; } }; i
int main(void)
{
san s1;
s1.print(60);
s1.print(23.26);
s1.print("Hello ");
return 0;
}
```

Operators Overloading

- You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.
- Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Syntax

class name operator operator-symbol (parameters);

Following is the list of operators which can be overloaded

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

Operators cannot be Overloaded

- ::
- .*
- .?
- :
- .

Two ways to write function body

Inside Class

- Create only one object and pass as argument subject to operator which have to overload

Outside Class

- Create two objects and pass as argument subject to operator which have to overload

Inside Class

```
class san
{
     int a;
    public:
      in()
    { cin>>a; }
 san  operator + (san s1);
   {
            a=a-s1.a;
}
out()
{ cout<<a;}
}
};
main()
{
      san s1,s2,s3;
      s1.in();
      s2.in();
      s3=s1+s2;
      s3.out();
}
```

Outside Class

```
class san
{
     int a;
    public:
      in()
    { cin>>a; }
 san  operator + (san s1);
out()
{ cout<<a;}
}

san  san  :: operator + (san  s1);
   {
             san s2;
              s2. a=a-s1.a;
}

};
main()
{
      san s1,s2,s3;
      s1.in();
      s2.in();
      s3=s1+s2;
      s3.out();
}
```