```
#include<stdio.h>
void f1(int a)
{
        printf("\n Inside int blcok");
}
void f1(float a)
{
        printf("\n Inside float blcok");
}
void f1(double a)
{
        printf("\n Inside double blcok");
}
int main()
{       f1(1);
                f1(1.2);
                f1(1.2f);
                f1((int)1.2);
                f1(1.2);
        return 0;
}
```

Function overloading :
Function having same name but differs either in different number of arguments or type of arguments or order of arguments such process of writing function is called function overloading .
Functions which is taking part in function overloading such functions are called as overloaded functions.

1.Function having same name but differs in number of arguments

```
eg:  void sum (int no1, int no2);
     void sum (int no1, int no2, int no3);
```

2.Function having same name and same number of arguments but differs in type of arguments

```
eg: void sum (int no1, int no2);
    void sum (int no1, double no2);
```

3. Function having same name ,same number of arguments but order of arguments are different

```
eg. void sum (int no1, float no2);
    void sum (float no1, int no2);
```

Name mangling: when we write function in c++ complier internally creates unique name for each and every function by looking towards name of the function and type of arguments pass to that function. Such process of creating unique name is called name mangling . That individual name is called mangled name.

Eg:

```
    sum (int no1, int no2, it no3)
                  sum@ int, int , int
    sum(int no1, int no2)
           sum@int,int
```

cin and cout

- C++ provides an easier way for input and output.
- The output:
  - cout << “Hello C++”;
- The input:
  - cin >> var;

```
#include<stdio.h>
int main()
{
    printf(" hellow world");
}
```

```
#include<iostream.h>
int main()
{
    cout<<"hellow world";
}
```

printf is a function & declared in stdio.h header file. So if we want to use printf it is necessary include stdio.h header file

cout is object of ostream class and ostream class is declared in iostream.h that's why if u want to use cout object is necessary to include iostream.h header file

operator which is used with cout is called as insertion operator <<

```
int res=30;
printf("res=%d", res);
```

```
int res=30;
cout<<"res="<<res;
```

```
int  a=10,b=5;
printf("a=%d b=%d",a,b);
```

```
int a=10, b=5;
cout<<"a="<<a<<" b="<<b;
```

```
#include<stdio.h>
int main()
{       int num;
         scanf("%d",&num);
}
```

```
#include<iostream.h>
int main()
{   int num;
     cin>>num;
}
```

scanf is a function & declared in stdio.h header file. So if we want to use scanf it is necessary include stdio.h header file

cin is object of istream class and istream class is declared in iostream.hthat's why if u want to use cin object is necessary to include iostream.h header file
operator which is used with cin is called as operator extraction (>>)

```
int  no1, no2;
scanf("%d%d",&no1, &no2);
```

```
int no1, no2;
cin>>no1>> no2;
```

What is class?

- **Building blocks that binds together data & code**
- **Program is divided into different classes**
- **Class has**
  - **Variables (data members)**
  - **Functions (member functions or methods)**

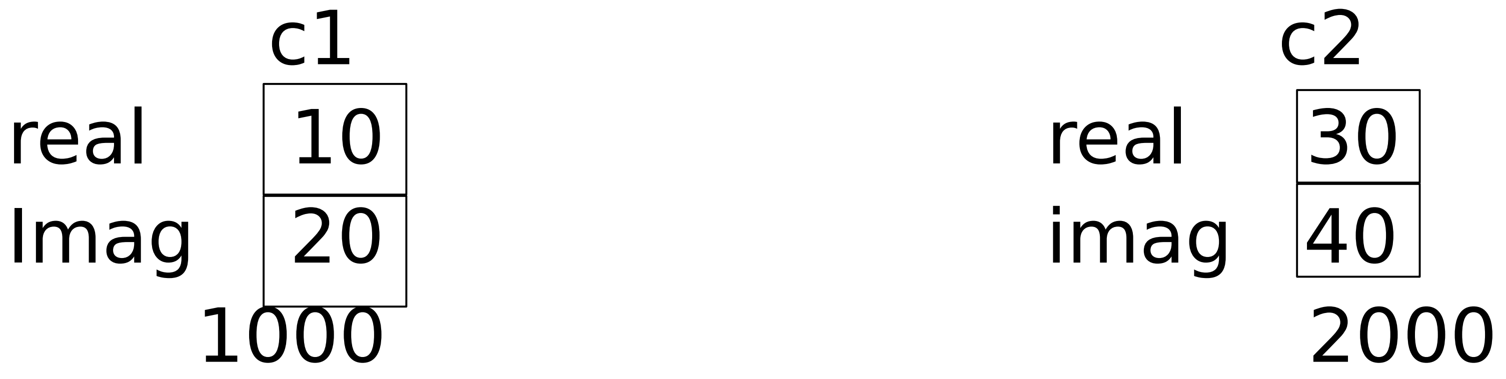
What is object?

- Object is an instance of class
- Entity that has physical existence, can store data, send and receive message to communicate with other objects
- Object has
  - Data members (***state*** of object)
  - Member function (***behavior*** of object)
  - Unique address(***identity*** of object)

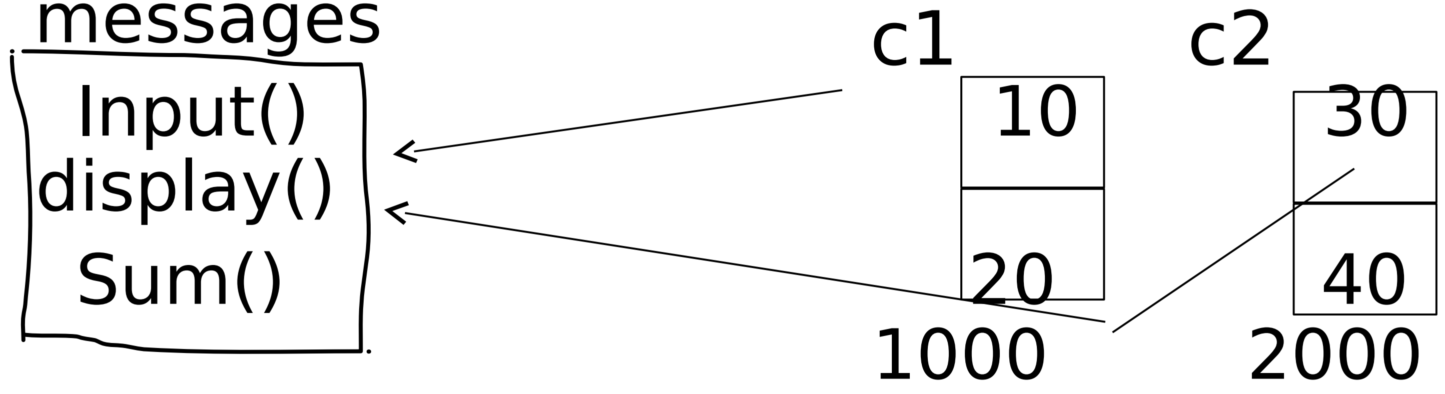
What is object?

- The values stored in data members of the object called as ‘state’ of object.
- Data members of class represent state of object.

TComplex c1, c2;



- Behavior is how object acts & reacts, when its state is changed & operations are done
- Behavior is decided by the member functions.
- Operations performed are also known as messages



Identity : Every object has a characteristics that makes object unique. Every object has unique address.

- *Identity of c1 1000 & c2 is 2000*

Class

- it is template or blue print for an object.
- object is always created by looking towards class, that’s why it is template for class.
- class is a logical entity.
- memory is not allocated to that class.
- class is collection of data members and member functions.

Object

- it is an instance of class.
- object is physical entity.
- memory is always allocated to object.

Data members

- **Data members of the class are generally made as private to provide the data security.**
- **The private members cannot be accessed outside the class.**
- **So these members are always accessed by the member functions.**

Member Functions

- **Member functions are generally declared as public members of class.**
- **Constructor : Initialize Object**
- **Destructor : De-initialize Object**
- **Mutators : Modifies state of the object**
- **Inspectors : Don’t Modify state of object**
- **Facilitator : Provide facility like IO**

Constructor

- **We can have constructors with**
  - **No argument : initialize data member to default values**
  - **One or more arguments : initialize data member to values passed to it**
  - **Argument of type of object : initialize object by using the values of the data members of the passed object. It is called as copy constructor.**

Copy Constructor

- TComplex c1(c2);
- This statement gives call to copy constructor. State of c1 become same as that of c2.
- If we don’t write, compiler provides default copy constructor.
- Generally, we implement copy constructor when we have pointer as data member which is pointing to dynamically allocated memory.

Constructor

- **Constructor is a member function of class having same name as that of class and don’t have any return type.**
- **Constructor get automatically called when object is created i.e. memory is allocated to object.**
- **If we don’t write any constructor, compiler provides a default constructor.**

Destructor

- **Destructor is a member function of class having same name as that of class preceded with ~ sign and don’t have any return type and arguments.**
- **Destructor get automatically called when object is going to destroy i.e. memory is to be de-allocated.**

Destructor

- **If we don't write, compiler provides default destructor.**
- **Generally, we implement destructor when constructor of the object is allocating any resource**
- **e.g. we have pointer as data member, which is pointing to dynamically allocated memory.**

this pointer

- **When we call member function by using object implicitly one argument is passed to that function such argument is called this pointer**
- ***this* is a keyword in C++.**
- **this pointer always stores address of current object or calling object.**
- **Thus every member function of the class receives *this* pointer which is first argument of that function.**
- **This pointer is constant pointer.**

For TCmplex class member function type of this pointer is TComplex * const this

classname * const this

- Size of object of empty class is always 1 byte
- When you create object of an class it gets 3 characteristics
- 1. State
- 2. Behavior
- 3. Identity
- When you create object of empty class at that time state of object is nothing. Behavior of that object is also nothing.but that object have unique identity(address).

memory in computer is always organized in form of bytes.

Byte is unit of memory. Minimum memory at objects unique address is one byte that's why size of empty class object is one byte.

Default arguments

Assigning default values to the arguments of function is called default arguments. Default arguments are always assigned from right to left direction.

Passing these arguments while calling a function is optional. If such argument is not passed, then its default value is considered. Otherwise arguments are treated as normal arguments

```
#include<iostream.h>
int sum (int a=0, int b=0, int c=0, int d=0)
{       return a+b+c+d;
}
int main()
{       int ans=sum();
        cout<<"sum="<<ans<<endl;
                ans=sum(10);
                cout<<"sum="<<ans<<endl;
                ans=sum(10, 20);
                cout<<"sum="<<ans<<endl;
                ans=sum(10, 20, 30);
                cout<<"sum="<<ans<<endl;
                ans=sum(10, 20, 30, 40);
                cout<<"sum="<<ans<<endl;
        return 0;
}
```

Inline functions

- **C++ provides a keyword *inline* that makes the function as inline function.**
- **Inline functions get replaced by compiler at its call statement. It ensures faster execution of function just like macros.**
- **Advantage of inline functions over macros: inline functions are type-safe.**
- **Inline is a request made to compiler.**
- **Every function may not be replace by complier , rather it avoids replacement in certain cases like function containing switch , loop or recursion may not be replaced**