

Structure

- Structure is a collection of similar or dissimilar data. It is used to bind logically related data into a single unit.
- This data can be modified by any function to which the structure is passed
- Thus there is no security provided for the data within a structure.
- This concept is modified by C++ to bind data as well as functions.

Diff Between struct in c & c++

- struct in c
- We can't write function inside structure
- At the time of creating variable of structure writing struct keyword is compulsory
eg. struct time t;
- By default all the members are accessible outside structure. C lang does not have a concept of access specifier
- If we want to call any function on structure variable
struct time t1;
input(&t1); print(&t1);
- struct in c++
- We can write function inside structure
- At the time of creating object of structure writing struct keyword is optional
eg.time t;
- By default all members of struct in c++ are public (we can make them private)
- If we want to call member function on object.
 - time t1;
 - t1.input();
 - t1.print();
 -

```
struct time {  
    int hr, min, sec;  
};  
  
void input( struct time *p)  
{  
    printf("Enter Hr Min Sec:");  
    scanf("%d%d%d", &p→hr,  
        &p→min, &p→sec);  
}  
struct time t;  
input(&t);
```

```
struct time  
{  
    int hr, min, sec;  
    void input()  
    {  
        printf("Enter Hr Min Sec::");  
        scanf("%d%d%d",&this→hr,  
            &this→min, &this→sec);  
    }  
};  
time t;  
t.input();
```

Access specifiers

- By default all members in structure are accessible everywhere in the program by dot(.) or arrow(→) operators.
- But such access can be restricted by applying access specifiers
 - private: Accessible only within the struct
 - public: Accessible within & outside struct

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 complex class member function type of *this* pointer is `Complex * const this`

`classname * const this`

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

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;
}
```



```
#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.

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);**

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++ compiler 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
```

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)

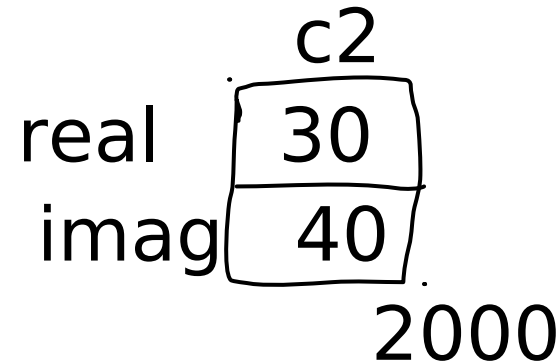
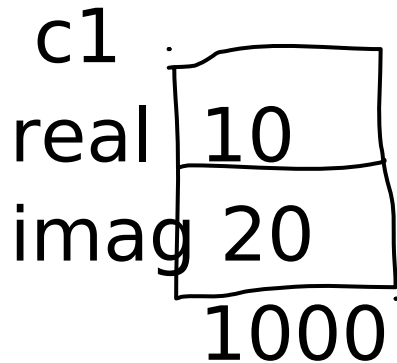
What is object?

- 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?

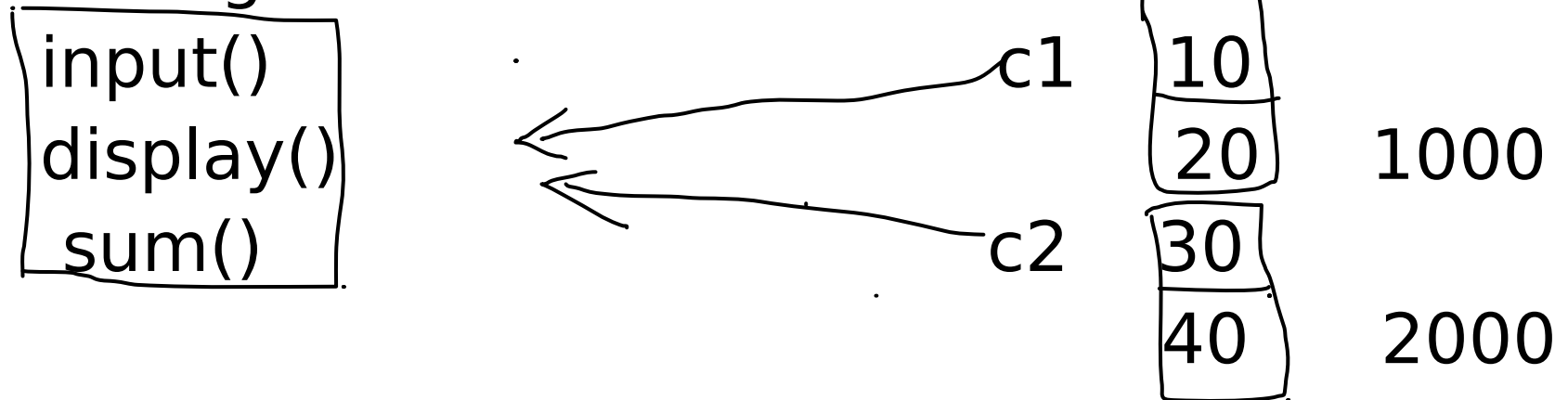
- The values stored in data members of the object called as 'state' of object.
- Data members of class represent state of object.

Complex c1, c2;



Behavior is how object acts & reacts,
when 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 it.

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.

Data types

- C++ supports all data types provided by C language. i.e. int, float, char, double, long int, unsigned int, etc.
- C++ add two more data types:
- 1. *bool* :- it can take *true* or *false* value. It takes one byte in memory.
- 2. *wchar_t* :- it can store 16 bit character. It takes 2 bytes in memory.

Variable declaration

- In C, variable should be declared at the start of the block.
- This restriction is removed in C++. We can declare the variables anywhere in function.

Comments in C++

- In C, comments are written as
- `/*This is comment*/`
- In C++, we can use above style. In addition C++ provides one more way for writing comments.
- `//This is comment`
- The second style is preferred for single line comments.