

## Class & Objects

- Class is an extension of structure in 'C'.
- But, a struct data type can not be treated like a built in data type.

e.g.

~~$C_3 = C_1 + C_2$~~  [is not possible]  
Illegal. [if all are of struct data type]

- Class provide facility to hide data [private, protected]
- Class can be inherit
- A class structure can have both variable & function as its member.
- Structure name [class name] can be used as another data type.

~ [Class: A class is a way to bind the data & its associated function together]

Specification: -

class class\_name {

private:

variable declaration;  
function declaration;

public:

variable declaration;  
function declaration;

} ; → termination

member

visibility of private member is restricted to class only

## creating objects :-

```
[ class-name Any-Name ; ]
```

## Accessing Class Member :-

object-Name . function\_name (actual-arguments)

\*\*\*

## Defining Member Function :-

A member function of a class is a function that has its definition @ its prototype within the class definition like any other variable.

Member function can be define within the class definition @ separately using (::) Scope resolution

e.g.

```
class item {
```

```
    int a;
```

```
    int b;
```

```
    Public :
```

```
        void getDate(int a, int b)
```

```
        {
```

```
            this->a = a;
```

```
            this->b = b;
```

```
        }
```

```
        void putDate(int a, int b)
```

```
        {
```

```
            cout << a << endl;
```

```
            cout << b << endl;
```

```
        }
```

```
    };
```

```
class item {
```

```
    int a;
```

```
    int b;
```

```
    Public :
```

```
        void getDate(int, int);
```

```
};
```

```
int item::getDate(int a, int b)
```

```
{
```

```
    a = a;
```

```
    b = b;
```

```
}
```

prototype  
declaration



- Declaring member function outside is as simple as declaring a normal function in C++ - you just need to use scope resolution with ~~not~~ class name.
- We can make member function inline.
- Inside class, • (dot) operator is not required to call a function. [Also called - Nested function]
- A member function can be declared private but its visibility will be limited to class only.
- The Array can be used as member variable in class

```
const int size = 10;
```

```
class Array {
```

```
    int a[size]; // It is allowed
```

```
public;
```

```
    void setval(void);
```

```
    void display(void);
```

```
};
```

### Memory Allocation for Objects ?

- When object of class is created, memory allocated at stack, but if some member ~~have~~ are pointer member [int \*a = new int] then that memory will be placed at heap.

## Static Data member!

- Initialize with zero (only one copy is created)
- visibility is limited to class but lifetime is entire program

Static member are known as class-variable because they are associated with entire class itself rather than with any class object

## Static member function

Static function can have access to only other static member.

- Static member f<sup>n</sup> can be called - class-name :: function-name

```
class item {  
    int code;  
    static int count;  
public:  
    void setCode (int a) {  
        code = a;  
    }  
    static void setCount (int a) {  
        count++; // count is static so ✓  
        code++; // code is not static ✗  
    }  
}
```

- Object of a class behave like any other data type so, Generally all activity of a data type is supported by object.



## Friend Function

Private member are not accessible outside the class but ~~function~~ friend function can have.

- It is not in the scope of the class to which it is declared.
- It can't be called using object but can be called directly.
- Friend function can not access class member directly rather it need object-name . member.
- It can be declared inside either in public or in private area.

class Sample {  
    int a;  
    int b;  
    Public:  
        void setData { a = 5 ; b = 10 ;  
};

How to declare a friend function  
(Just add friend prefix)

friend float mean(Sample s);  
          ↑          ↑          ↓          ↓  
          Prefix  Return type  fn name  parameter

```
{  
    float mean(Sample s) {  
        return (s.a + s.b) / 2.0 ;  
    }
```

```
int main() {  
    Sample X // object X  
    X.setData // calling a Normal function  
    mean(X) // calling a friend function  
}
```

## Const member function :-

- If our function is not altering any data in the class then we may declare it ~~const~~ const.

Void      mul ( int , int )      const ;

↓                      ↓                      ↓                      ↓

Return type      Fun Name      Parameter      constant type.

## Pointer to member

```
class A {  
    Private :  
        int m;  
    Public :  
        void show();  
};
```

A::\* ip = &A::m;

↓                      ↓

Pointer to member of "A" class      Address of the "m" member of "A" class.

[ ip is created & it act like class member ]

→ let say a object of class "A" is a  
the a.\*ip → Access the member m.