creating objects :-

[class_name Any_Name ;]

Accessing Class member :-

Object_Name . fnction_name (actual_arguments)

Defining Member function :-

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

[member fnction can be define within the class definition (or) seperately using (::) scope resolution]

e.g.

```
class item
{
    int a;
    int b;
public:
    void getdata(int a, int b)
    {
    this->a = a;
    this->b = b;
    }
    void PutData(){
    cout << a << endl;
    cout << b << endl;
    }
};
```

```
class item {
    int a;
    int b;
public:
    void getdata(int, int);
};

int item :: getdata (int a, int b){
    a = a
    b = b;
}
```

prototype declaration

- Defining member function outside is as simple as defining a normal function in c++ - you just need to use scope resolution with name 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 setVale(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 fuction

- Static fuction can have access to only other static member.
- Static member f^n can be called - class-name :: fuction-name

```
class item{
        int code;
        static int cout;
        public:
            void setCode(int a){
                code = a;
            }
            static void setCout(){
                cout++; // cout 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 fuction

Private member are not accessible outside the class but ~~fuction~~ friend fuction 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 fuction can not access class member directly rather it need (object-name . member).
- It can be declared ~~inside~~ either in public or in private are.

```
class Sample {
    int a;
    int b;
public:
    void setData() { a = 5; b = 10; }
    friend float mean(Sample S);
};
```

How to declare a friend fuction → (just add friend prefix)

friend: Prefix; float: Return type; mean: fu Name; (Sample S): parameter

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

```
int main() {
    Sample X   // object X
    X.setValue // calling a Normal fuction
    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 — Fn Name — Parameter — constant type.

Pointer to member

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

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

A::* → Pointer to member of "A" class

&A::m → 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

then a.*ip ⟶ Access the member m.