static variables are also called class variable as for every class they are shared. We can also access them like, {{class name}} :: static var;

class as and {{object name}}. static var; but first one is recommended.

int <class name> :: y = 10;

Also note that by default static variable has value of '0' initialize in it.

→ Static function :-

~~Static member function are made in class~~ can

static member function only access static data member and other static function, they can't access other private data members and functions.

Also static function can be called without making object. Directly with class. like

{{class name}} :: static fun();

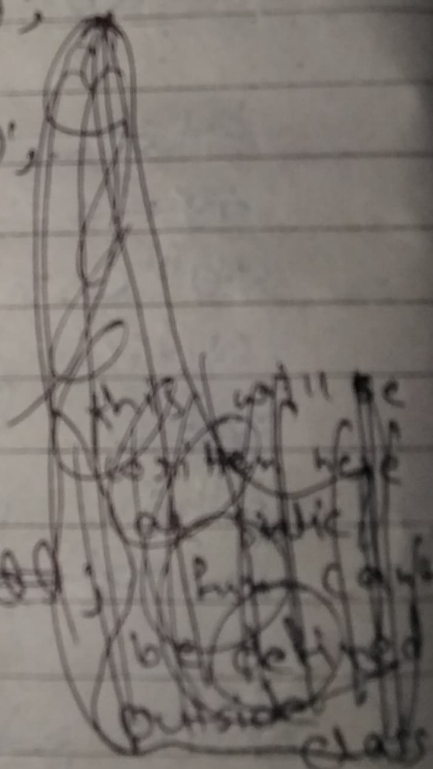
We can't use this pointer with static function.

```
#include<iostream>
using namespace std;

class x
{
    char name[20];
    int marks;
static int count;
public:
   static void counter();
     void getdata();
     void printdata();
};

void x :: counter()
{
   ++count;
   cout<<" count "<<endl;
}

void x :: getdata()
{
   cout<<" Put student name"<<endl;
   cin>> name;
   cout<<" Put student marks"<<endl;
```



```
    cin>>marks;
}

void x::printdata()
{   cout<<count;
    cout<<" student name is"<<endl;
    cout<<name<<endl;
    cout<<" student marks is"<<endl;
    cout<<marks<<endl;
    cout<<endl<<endl;

}

int x::count;

int main()
{
    x std1, std2, std3;

    x::counter();
    std1.getdata();
    std1.printdata();

    x::counter();
    std2.getdata();
    std2.printdata();
```

```
    X::Counter();
std3.getdata();
 std3.printdata();

 return 0;

}
```

We all know class is a user-defined data-type or you can say a coustum data type.

Like datatype, ~~class~~ what we can do with datatype we can do all things with class.

like defined a variable?

Yes, you can define a variable with class ~~which~~ and ~~are~~ we are doing it, we call these variable, defined with class as a datatype, Object

So, for comparision sake we can assume object as variable.

Now, if we can make arrays of variable, so, can we do like this with objects?

Yes, we can, we can make arrays of objects. Which will helps us to write names of many objects one by one.

Also we can access each objects as we asses each datapiece in an array of variable by using index. -

ex. -

Q. WAP to take input name, std and marks of 10 students and print them.

```
#include<iostream>
using namespace std;

class student
{
    int std;
    char name[20];
    int mark;

public:
    void indata(void);
    void pntdata(void);
};

void student :: indata( )
{

    cout<<"Input student name"
                <<endl;
    cin>> name;

    cout<<" Input student std"<<
                endl;
    cin>> std;
```

```
    cout<<" Input student marks"
                        <<endl;
    cin>> mark;
}

void student :: prntdata()
{
    cout<" student name, std and
        marks are given below"
                    <<endl;
    cout<<name<<endl<< std<<
        endl<<mark<<endl
            <<endl;
}

int main()
{
    student  std[10];
    for(int i=0; i<10; i++)
    {
        std[i].indata();
    std[i].prntdata();
}
```

```
    return 0;
};

int main()
{
    student stnd[10];

    for(int i=0; i<10; i++)
    {
        stnd[i].indata();
    }
    for(int i=0; i<10; i++)
    {
        stnd[i].printdata();
    }

    return 0;
}
```

Since, we have assumed object as variable of class datatype, as variable we can pass object as function arguments too.

```
#include<iostream>
using namespace std;

class SumC
{
    int a,b;

  Public:
    void sum(int v1, int v2)
    {
      a = v1;
      b = v2;

    }
    void sum (sumcO1, sumcO2)
    {
       a = O1.a + O2.a;
       b = O1.b + O2.b;
    }
```

```
void print (void)
{
  cout<<"sum of a and is"
      <<a<<"+"<<b<<"=";
      <<endl;
}

int main()
{
  sumC a1, a2, a3;

  a1.sum(1,2);
  a1.print();

  a2.sum(5,7);
  a2.print();

  a3.sum(a1, a2);
  a3.print();

  return 0;
}
```

here in this program we have also used function overloading.

void sum(int v_1, int v_2)
void sum(sumC O_1, sumC O_2)

→ Friend functions :-

Friend function is any non-member or forign function, which have the access of a given class.
private data members
It is not a member function of the class, so, it can't be called with the help of the object. But it can called directly in main
for ex.-

object_name.friend_function();

is invalid statement.

Usually it contains objects as the argument as friend function needs object of the class to access

private data members of the class.
for ex-

Object_name . private_data_members;

Demonstration programm :-

```
#include <iostream>
using namespace std;

class complex
{
    int a,b;
    friend complex Sumcomplex
            (complex o1, complex o2);

/* syntax to declair friend function

friend <return type> <friend function>
    (argument );

Also note that friend function
can be declared within
public or private doesn't matter */
```

public:

```
void setnumber(int n1, int n2)
{
    a=n1;
    b=n2;
}

void printnumber()
{
cout<<" Your number is"<<a<<"+"
    << b <<"i" <<endl;
}
};

Complex sumComplex(complex o1, complex o2)
{
    Complex o3;
    o3.setnumber((o1.a+o2.a),
                 (o1.b+o2.b));
    return o3;
}
```

x=8

(1,4) (5,8)

```
int main()
{
    complex C1, C2, sum;

    C1.setnumber(1,4);
    C1.printnumber();

    C2.setnumber(5,8);
    C2.printnumber();

    sum = sumcomplex(C1, C2);

    Sum.printnumber();

    return 0;
}
```

Till now, we have learned how to make a function, a friend function. But the function which we made friend with, a normal function.

What if we want to make ~~another~~ function of another class a friend

function. For that you have do something extra than making a normal ~~non-member~~ function ~~of any cls~~ which is not in any class a friend function.

we can understand it by this example.

```
#include<iostream>
using namespace std;

class Complex; // Forward declaration
// Forward declration is needed because
// we are gonna use this class name
// before its defination. Also
// Forward declaration tells us that
// compilar will get a class with
// this name.

class Calculator
{
  Public:
     int add(int a, int b)
     {
     }  return (a+b);
```

```
    int sumReal_complex(complex, complex);
    int sumComp_complex(complex, complex);
};

class complex
{
    int a, b;
    friend int calculator::sumRealcomplex
                        (complex, complex);

    friend int calculator::sumCompcomplex
                        (complex, complex);

public:
    void setnumber(int n1, int n2)
    {
        a=n1;
        b=n2;
    }
    void printnumber()
    {
    cout<<" a°<<" + "<<b<<"i"<<endl;
};  }
```

```
int calculator :: sumrealcomplex(complex o1,
                                 complex o2)
{
    return (o1.a + o2.a);
}

int calculator :: sumcompcomplex(complex o1,
                                 complex o2)
{
    return (o1.b + o2.b);
}

int main()
{
    complex o1, o2;
    o1.setnumber(1,4); o1.printnumber();
    o2.setnumber(5,7); o2.printnumber();
    calculator calC;
    int res = calC.sumrealcomplex(o1, o2);
    cout<<"Sum of real parts are"<<res<<endl;
```

```
int resc = calC.SumcompComplex(O1,O2);
Cout<<"The sum of complex part are"
    <<resc<<endl;

  return 0;
}
```

Now, in this programm you have seen forward declaration of class, it tells compiler that a class of the given name will be defined later, but you can use its name before ~~declaration~~ definition.

now, we have another class calculator which have two function SumRealcomplex() and SumCompComplex() who want to access private data of another class complex.

We have not defined them in calculator class but outside, by just declaring them in calculator as we can't define them in

Calculator as their defination need/ contain name of private data members of complex class which is not defined. We have already told compiler that a complex class will be defined later but how can we use name of data members before defination. So we need to define those function after defining complex class.

→ Friend class :-

We have seen how to make a function of another class friend function of another class. But if we have more functions in a class and we want that they can access private data of another class, we can simply make them friend function of another class, but since they are many in numbers, we just can't write the same code of friend

function.

What if, we can make the whole class a friend class, so that any function of that class can access private data of this class.

we can do so, just by writing

friend class <name of class>
who want to be friend.

example programm for this one is same as privious one just a simple change is defining complex class.

```
Class complex
{
    int a,b;
    friend class calculator;
    void setnumber(int n1, int n2)
    { a=n1;
    } b=n2;
```

```
    void printnumber()
    {
      cout<<a<<" +"<<b<<" i"<<endl;
    }
};
```

We have learnt how to make a function friend function who is using private member of one class, what if a function is accessing private data memeber of more than one class.

```
#include<iostream>
using namespace std;

class Y;
class X
{
    int a;
public:
    friend void add(X,Y);
    void getdata(void);
};
```

```
class Y
{
    int b;
public:
    friend void add(X,Y);
    void setdata(void);
};

void add(X o1, Y o2)
{
    cout<<" sum
```

```
void X::getdata()
{
    cout<<" Input an integer "<<endl;
    cin>> a;
}

void Y::setdata()
{
    cout<<" Input another integer "
        <<endl;
    cin>> b;
}
```

```
void add(XO1, YO2)
{
    cout<<" sum of a and b is"
        <<(O1.a + O2.b) <<endl;

}

int main()
{

    XO1;
    O1.getdata();

    YO2;
    O2.setdata();

    add(O1, O2);
    return 0;
}
```