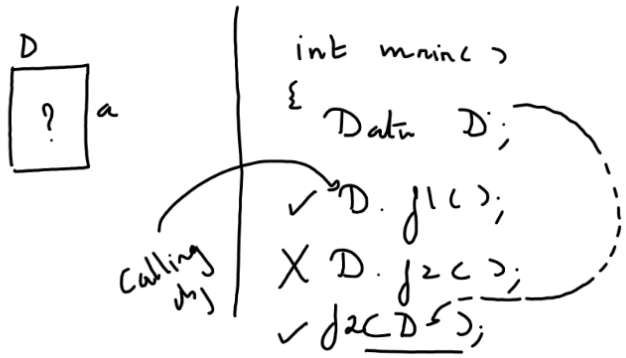
```
class Data
{
  int a;
public:
        void f1();
        friend void f2(Data);
};
```



Properties Of Friend Function
==========

2. Whenever we define a friend function , netither the class name nor the SCOPE RES OPERTOR is applied. Moreover the keyword friend also is not written while defining a friend function/

3. Whenever we call a friend function , we never use any CALLING OBJECT or DOT OPERATOR . It only accepts the object as argument whose data it wants to access.

4. It doesn't matter in which section of the class we have declared a friend function because we can always access it from anywhere in our program.

5. Since friend functions do not have any calling object , so we cannot use "this" pointer in the body of a friend function

```
#include <iostream>
using namespace std;
class Student
{
    int roll;
    char grade;
    float per;
public:
    void get();
    friend void show(Student);
};
void Student::get()
{
    cout<<"Enter roll.grade and per:";
    cin>>roll>>grade>>per;
}
void show(Student P)
{
cout<<P.roll<<","<<P.grade<<","<<P.per<<endl;
}
```

```
int main()
{
    Student S;
    S.get();
    show(S);
    return 0;
}
```

A function declared as friend of 2 classes

==========

```
#include <iostream>
using namespace std;
class Beta;
class Alpha
{
   int a;
public:
   void get()
   {
      cout<<"enter a:";
      cin>>a;
   }
   friend void compare(Alpha,Beta);
};
class Beta
{
   int b;
public:
   void set()
   {
      cout<<"enter b:";
      cin>>b;
   }
   friend void compare(Alpha,Beta);
```

```
void compare(Alpha obj1,Beta obj2)
{
   if(obj1.a>obj2.b)
      cout<<"Gr is "<<obj1.a<<endl;
   else if(obj2.b>obj1.a)
      cout<<"Gr is "<<obj2.b<<endl;
   else
      cout<<"Both are equal";
}
int main()
{
   Alpha obj1;
   Beta obj2;
   obj1.get();
   obj2.set();
   compare(obj1,obj2);
   return 0;
}
```