Friend Functions

Now, we'll take a look at a special category of functions called friends.

So far we have observed that the private data members of a class are only accessible through the functions present in that class. Nothing from outside can manipulate the class object without using its functions.

What if we need to access class variables in a function which is not a part of the class? That function would have to become a **friend** of the class.

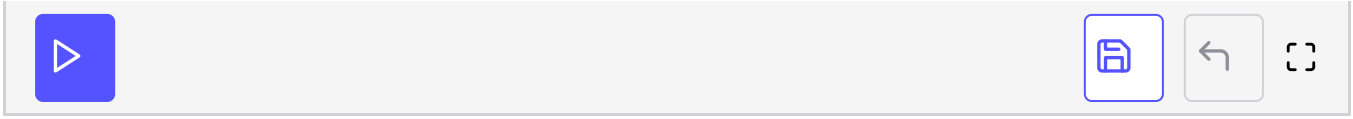
> A friend function is an independent function which has access to the variables and methods of its befriended class.

To create a friend function for a class, it must be declared in the class along with the `friend` keyword.

Let's create a `Ball` class to explain this better:

```
#include <iostream>
#include <string>

using namespace std;

class Ball{
  double radius;
  string color;

  public:
  Ball(){
    radius = 0;
    color = "";
  }

  Ball(double r, string c){
    radius = r;
    color = c;
  }

  void printVolume();
  void printRadius();
```

```
  // The friend keyword specifies that this is a friend function
  friend void setRadius(Ball &b, double r);

};

// This is a member function that calculates the volume.
void Ball::printVolume(){
  cout << (4/3) * 3.142 * radius * radius * radius << endl;
}

void Ball::printRadius(){
  cout << radius << endl;
}

// The implementation of our friend function
void setRadius(Ball &b, double r){
  b.radius = r;
}

 int main(){
   Ball b(30, "green");
   cout << "Radius: ";
   b.printRadius();
   setRadius(b, 60);
   cout << "New radius: ";
   b.printRadius();
   cout << "Volume: ";
   b.printVolume();
 }
```



In line 25, we can see that the `Ball` object is being passed by reference to the friend function. This is a crucial step in the functionality of the friend. If the object is not passed by reference, the changes made in the friend function will not work outside its scope. Basically, our object will not be altered.

The `setRadius()` function is completely independent of the `Ball` class, yet it has access to all the private variables. This is the beauty of the `friend` keyword.

This concludes our discussion on the basics of the classes in C++. The next section deals with the concept of data hiding, which plays a pivotal role in implementing efficient classes.