constant member function

Constant member functions are those functions that are denied permission to change the values of the data members of their class. To make a member function constant, the keyword const is appended to the function prototype and also to the function definition header.

Example:

```
class my_Class {
        int x;
public:
        void set_data(int a) {
           x = a;
        }
        int get_data() const {
                ++x;
                return x;
        }
};
main(void)
{
        my_Class obj;
        obj.set_data(10);
        cout << obj.get_data();
}
```

Output:

error: increment of member 'my_Class :: x' in read-only object

constant objects

Like member functions and member function arguments, the objects of a class can also be declared as const. An object declared as const cannot be modified and hence, can invoke only const member functions as these functions ensure not to modify the object. A const object can be created by prefixing the const keyword to the object declaration. Any attempt to change the data member of const objects results in a compile-time error.

Example1:

```
class my_Class {
public:
   int value=10;
        void display() const {
                cout << value << " const method() invoked" << endl;
        }
};
int main(void) {
        const my_Class obj;
        obj.display();
}
```

A const member function can be called by any type of object. Non-const functions can be called by non-const objects only.

Example2:

```
class my_Class {
public:
        int value=10;
        void display() {
                cout << "const method() invoked" << endl;
        }
};
int main(void) {
        const my_Class obj;
        obj.display();
}
```

Output,

```
error: passing 'const my_class' as ‘this’ argument discards qualifiers [-fpermissive]
```