CodInClub

Powered By BridgeLabz

Wk6 /S5/ Lecture # : DSOOPS-29

Topics Covered

- Constant Member Functions (`const` after function signatures)
- Constant Objects (`const` objects)
- Friend Functions
- Friend Classes
- Practice Problems (2 Easy, 1 Medium, 1 Hard)

Constant Member Functions

A constant member function promises *not to modify* any member variables of the object it's called on.
This is done by adding `const` at the end of the function declaration and definition, like so:

```
class Point {
    int x, y;
public:
    Point(int a, int b): x(a), y(b) {}
    int getX() const { return x; }    // Constant member function
    void setX(int val) { x = val; }   // Non-const, can modify
members
};
```

Key Rules:

- Inside a const member function, you cannot modify any data member (except those marked as `mutable`).
- Const member functions *can* be called on `const` objects.

Why use it?

- Shows which functions are safe (do not change the object).
- Improves code readability and enables usage with const objects.

Constant Objects

A constant object is an object declared with the `const` keyword, making all its data members *read-only* through that object (except those marked as `mutable`).

- Only const member functions can be called on a const object.

```
class Student {
public:
    std::string name;
    int getLength() const { return name.length(); }
};

int main() {
    const Student s = {"Alice"};
    // s.name = "Bob";      // Error: cannot modify member of
const object
    std::cout << s.getLength() << std::endl; // OK: getLength()
is const
    return 0;
}
```

Friend Functions

A friend function is a function (not a member of a class) but is allowed access to the class's private and protected members.

- Declared inside the class with the `friend` keyword.
- Useful for operator overloading or functions that need special access.

Example:

```
class Box {
private:
    int secret;
public:
    Box(int s): secret(s) {}
    friend void revealSecret(const Box& b);
};
void revealSecret(const Box& b) {
    std::cout << b.secret << std::endl;  // Allowed: friend can
access private members
}
```

Friend Classes

A friend class is a class that is given access to another class's private and protected members.

- Declared inside the class using: `friend class ClassName;`
- All member functions of the friend class get access.

Example:

```
class Engine; // Forward declaration


class Car {
private:
    int speed;
public:
    Car(int s) : speed(s) {}
    friend class Engine;  // Engine is a friend class
};

class Engine {
public:
    void printSpeed(const Car &c) {
        std::cout << "Speed = " << c.speed << std::endl; // OK:
can access private 'speed'
    }
};
```

Practice Problems and Activities

Easy 1

What's wrong with the code? Correct it so `getVal()` can be called on the constant object:

```
class A {
    int val;
public:
    int getVal() { return val; }
};

int main() {
    const A a;
    std::cout << a.getVal() << std::endl;
    return 0;
}
```

Hint: What's missing in getVal()?

Easy 2

Add a friend function `showSecret` to the following class so it can print the private member `code`:

```
class Vault {
private:
    int code;
public:
    Vault(int c) : code(c) {}
    // Friend function declaration here
};
```

Write the friend function and show its use in main.

Medium

Given two classes `Alpha` and `Beta`, make Beta a friend of Alpha so that Beta's function can access Alpha's private data. Implement accordingly and demonstrate usage.

Hard

Suppose you have a class `Student` with private marks and a class `Teacher` that needs to set a Student's marks.

1. Make appropriate use of friend function or friend class (choose which is best).
2. Provide code to show that Teacher can update Student's marks, but main cannot do so directly.
3. Explain briefly why you would choose friend function vs. friend class here.

Wrap-Up & Key Takeaways

- Const member functions promise not to modify the object; only these can be called on `const` objects.
- Const objects cannot have their data members changed (except `mutable`), and only const functions are allowed.
- Friend functions and friend classes provide controlled access to private data for trusted code, but should be used sparingly (to maintain good encapsulation).
- These concepts are vital for robust, safe, and expressive class design in C++.