📘 Getters and Setters in C++

🧠 What Are They?

In C++, **getters** and **setters** are special member functions used to **access** and **modify** private data members of a class. They are part of the **encapsulation** principle in object-oriented programming, which promotes **data hiding** and **controlled access** to class members.

🔒 Why Use Them?

1. **Encapsulation**: Keeps internal data safe from unintended interference.
2. **Validation**: Setters can include logic to validate data before modifying a member.
3. **Read-only or Write-only Access**: You can expose only a getter or a setter.
4. **Abstraction**: Hides implementation details from the user.

🧱 Syntax and Example

Example Class: `Student`

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

class Student {
private:
    string name;
    int age;

public:
    // Constructor
    Student(string n, int a) {
        name = n;
        setAge(a); // Use setter to apply validation
    }

    // Getter for name
    string getName() const {
        return name;
    }

    // Setter for name
    void setName(const string& n) {
        name = n;
    }

    //
    //ALTERNATIVE naming: do NOT use the 'get' or 'set' prefix
    //
    // Getter for age
    int Age() const {
        return age;
    }

    // Setter for age with validation
    void Age(int a) {
        if (a > 0 && a < 150) {
            age = a;
        } else {
            cout << "Invalid age. Setting to default (18)." << endl;
            age = 18;
        }
    }
};
```

🪄 Usage

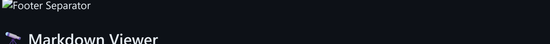
```cpp
int main() {
    Student s("Alice", 20);

    // Accessing data using getters
    cout << "Name: " << s.getName() << endl;
    cout << "Age: " << s.Age() << endl;

    // Modifying data using setters
    s.setName("Bob");
    s.Age(200); // Invalid, triggers validation

    cout << "Updated Name: " << s.getName() << endl;
    cout << "Updated Age: " << s.Age() << endl;

    return 0;
}
```

🧩 Notes

- **Const correctness**: Getters are often marked `const` because they don't modify the object.
- **Inline functions**: Getters and setters can be defined inline for simplicity.
- **Performance**: Modern compilers optimize simple getters/setters well, so there's usually no performance penalty.

🎯 Quiz!

Here's a short quiz on the topic: quiz



🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 7
- Day 8
- Day 9

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 7
- Day 8
- Day 9

Weekly Topics

Here are the topics for the week...

- Classes
- Structs
- Fields
- Getters and Setters
- Constructors
- Instances
- Inheritance
- Polymorphism
- Pointers
- Upcasting
- Misc. Concepts
- 4 Pillars of OOP