Using `const` in C++

- The '`const`' keyword enforces immutability and helps make code safer.
- It can be used with variables, pointers, functions, and objects.

1. Constant Variables

- Example: `const int x = 10;`
- x cannot be modified after declaration.
- Trying to assign x = 5 will cause a compiler error.

2. Pointers with const

- There are three main uses:
- a) `const int* ptr = &x; // Can't modify *ptr`
- b) `int* const ptr = &x; // Can't change ptr`
- c) `const int* const ptr = &x; // Neither *ptr nor ptr can change`

3. Const Function Parameters

- Used to prevent modification of passed-in values.
- `void printValue(const int value);`
- `void printRef(const int& value); // Read-only reference`
- `void display(const std::string& str); // Safe and efficient`

4. Const Member Functions

- Indicates the method doesn't modify the object.
- `class MyClass { int getValue() const; };`
- Modifying members inside a const method is not allowed.

5. Const Objects

- `const MyClass obj;` // Object is immutable
- Only const member functions can be called on it.

6. Constexpr vs Const

- `const`: value can’t change after initialization.
- `constexpr`: value is known at compile time and is also const.
- Example: `const int x = 10;` vs `constexpr int y = 20;`