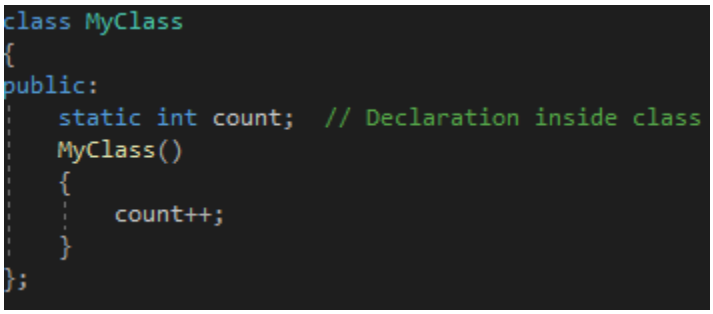
Object Oriented programming and software engineering – Lab 14

Adam Korytowski – 2025

1. Static

In C++, a static member variable is shared by all instances of a class, not duplicated per object. This means that no matter how many objects of the class you create, there is only one copy of the static variable stored in memory. It is useful for keeping track of information common to all objects, like a counter for how many objects were created. Static member variables must be defined outside the class body (even though they are declared inside) because they have to be allocated independently of any object.

Example of use – *static* keyword for counting objects of a class:



```
class MyClass
{
public:
    static int count;  // Declaration inside class
    MyClass()
    {
        count++;
    }
};
```

```
MyClass a, b, c;
std::cout << "Number of objects: " << MyClass::count << std::endl;  // Output: 3
```

Now, suppose you have a class where all objects should share a common setting, like a global configuration (e.g., a tax rate, exchange rate, or database connection string).

Example of use – common setting of each object:

```
class Product
{
private:
    double price;
    static double taxRate;  // shared by all products

public:
    Product(double p) : price(p) {}

    double getPriceWithTax() const
    {
        return price * (1 + taxRate);
    }

    static void setTaxRate(double rate)
    {  // update tax rate for everyone
        taxRate = rate;
    }
};

// Define and initialize static member outside
double Product::taxRate = 0.1;  // 10% tax
```

```
Product p1(100);
Product p2(200);

std::cout << "Price with tax (p1): " << p1.getPriceWithTax() << std::endl;  // 110
std::cout << "Price with tax (p2): " << p2.getPriceWithTax() << std::endl;  // 220
```

A static member function belongs to the class itself, not to any particular object. It can be called even when no object of the class exists. Because it does not operate on an object, a static function cannot access non-static member variables or this pointer – it can only work with static data members or external variables passed to it. Static functions are often used for utility tasks or operations that logically belong to the class but don't need object-specific data. Class itself and can be called without creating an object.

```cpp
class Math
{
public:
    static int add(int a, int b)
    {
        return a + b;
    }
};

int main()
{
    std::cout << "Sum: " << Math::add(5, 7) << std::endl;  // Output: 12
}
```

2. Const

The const keyword in C++ is used to protect data from being modified accidentally. You can apply const to variables to make them read-only, to function parameters to prevent the function from changing the input, to member functions to guarantee they won't modify the object, and to pointers to control whether you can change the pointer itself or the data it points to. Using const correctly makes your code safer, clearer, and easier to debug.

```cpp
const double pi = 3.14159;
pi = 3.14;  // ✕ Error: cannot modify a const variable
```

- Use const in function parameters to prevent accidental modification of the input, especially for large objects passed by reference.

```cpp
void printName(const std::string& name)
{
    std::cout << "Name: " << name << std::endl;
}
```

- Const Member Function – declare a member function as const to promise that it will not modify any data members of the class.

```cpp
class Player
{
private:
    std::string name;
public:
    Player(std::string n) : name(n) {}
    std::string getName() const
    {  // const after parentheses
        return name;
    }
};
```

3. Tasks (2 pts each)

- Add a static member variable for the purpose of counting objects of your class that keeps track of how many objects have been created. Increment the counter inside the constructors of the derived classes. Write a static function getObjectCount() to return the total number of objects of your class created so far
- In one of your classes, add a static function that returns a default value of one of your class parameters. Call the function without creating an object and print the result inside main()
- Modify one of your classes so that one of its methods takes a const argument to ensure the parameter cannot be changed inside the function.
- Mark one of the getter function in one of your classes as const, ensuring it does not modify the object's internal state