# 7. Constructors and Destructors

## 7.1 Constructors

**Purpose:** Special member function that initializes an object when it's created.

**Types of Constructors:**

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

class Student {
private:
    string name;
    int id;
    float gpa;

public:
    // 1. Default Constructor
    Student() {
        name = "Unknown";
        id = 0;
        gpa = 0.0;
        cout << "Default constructor called" << endl;
    }

    // 2. Parameterized Constructor
    Student(string n, int i, float g) {
        name = n;
        id = i;
        gpa = g;
        cout << "Parameterized constructor called" << endl;
    }

    // 3. Constructor with Default Parameters
    Student(string n, int i = 0, float g = 0.0) {
```

```
        name = n;
        id = i;
        gpa = g;
    }

    // 4. Copy Constructor
    Student(const Student& other) {
        name = other.name;
        id = other.id;
        gpa = other.gpa;
        cout << "Copy constructor called" << endl;
    }

    void display() {
        cout << "Name: " << name << ", ID: " << id << ", GPA: " << gpa << endl;
    }
};

int main() {
    Student s1;                          // Default constructor
    Student s2("Alice", 101, 3.8);       // Parameterized constructor
    Student s3 = s2;                     // Copy constructor
    Student s4(s2);                      // Copy constructor (explicit)

    s1.display();
    s2.display();
    s3.display();

    return 0;
}
```

# 7.2 Member Initializer List

**Preferred way to initialize member variables:**

```
class Rectangle {
private:
    double width;
    double height;
```

```
    const int id;  // const members must use initializer list

public:
    // Using initializer list (more efficient)
    Rectangle(double w, double h, int i) : width(w), height(h), id(i) {
        // Constructor body
    }

    // Alternative (less efficient - assigns after construction)
    // Rectangle(double w, double h) {
    //     width = w;
    //     height = h;
    // }
};
```

# 7.3 Destructors

**Purpose:** Special member function called when an object is destroyed, used for cleanup.

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

class FileHandler {
private:
    string filename;
    bool isOpen;

public:
    // Constructor
    FileHandler(string fname) : filename(fname), isOpen(false) {
        cout << "Opening file: " << filename << endl;
        isOpen = true;
    }

    // Destructor
    ~FileHandler() {
        if (isOpen) {
            cout << "Closing file: " << filename << endl;
            isOpen = false;
```

```
        }
    }

    void writeData(string data) {
        if (isOpen) {
            cout << "Writing to " << filename << ": " << data << endl;
        }
    }
};

int main() {
    {
        FileHandler file1("data.txt");
        file1.writeData("Hello World");

        // Destructor automatically called when file1 goes out of scope
    }

    cout << "After block" << endl;

    return 0;
    // Destructor called for any remaining objects
}
```

**Output:**

```
Opening file: data.txt
Writing to data.txt: Hello World
Closing file: data.txt
After block
```

---

Revision #1
Created 2025-11-09 13:21:27 UTC by DS
Updated 2025-11-09 13:23:40 UTC by DS