Tutorial 30 - Parameterized and Default Constructors in C++

Key Concepts

1. **Default Constructor**:
   - Takes no parameters.
   - Initializes data members with default values.
   - Called automatically when an object is created without arguments.
2. **Parameterized Constructor**:
   - Takes one or more parameters.
   - Allows initialization of objects with specific values at the time of creation.

Code Examples and Explanation

Example 1: Parameterized Constructor

Code Snippet 1:

```cpp
#include<iostream>
using namespace std;
class Complex {
    int a, b;
public:
    Complex(int, int); // Constructor declaration
    void printNumber() {
        cout << "Your number is " << a << " + " << b << "i" << endl;
    }
};
Complex::Complex(int x, int y) { // Parameterized Constructor
    a = x;
    b = y;
}

```

Explanation:

1. **Class** `Complex`:
   - Private members: `a`, `b`.
   - Constructor accepts two parameters and assigns them to `a` and `b`.
   - Function `printNumber()` displays values.
2. **Main Program (Snippet 2)**:

```cpp
int main() {
    // Implicit call
    Complex a(4, 6);
    a.printNumber();
    // Explicit call
    Complex b = Complex(5, 7);
    b.printNumber();
    return 0;
}

```

Execution:

- **Object** `a`: Implicitly calls the constructor with `(4, 6)`.
- **Object** `b`: Explicitly calls the constructor with `(5, 7)`.
- Prints:

```
Your number is 4 + 6i
Your number is 5 + 7i

```

Example 2: Parameterized Constructor (Points)

Code Snippet 3:

```cpp
#include<iostream>
using namespace std;
class Point {
    int x, y;
public:
    Point(int a, int b) { // Parameterized Constructor
        x = a;
        y = b;
    }
    void displayPoint() {
        cout << "The point is (" << x << ", " << y << ")" << endl;
    }
};

```

Explanation:

1. **Class** `Point`:
   - Private members: `x`, `y`.
   - Constructor initializes `x` and `y` with passed values.
   - Function `displayPoint()` prints coordinates.
2. **Main Program (Snippet 4)**:

```cpp
int main() {
    Point p(1, 1);  // Implicit call with (1, 1)
    p.displayPoint();
    Point q(4, 6);  // Implicit call with (4, 6)
    q.displayPoint();
    return 0;
}

```

Execution:

- **Object** `p`: Initializes with `(1, 1)` and prints `The point is (1, 1)`.
- **Object** `q`: Initializes with `(4, 6)` and prints `The point is (4, 6)`.

Short Notes for Notebook

Parameterized & Default Constructors

1. **Default Constructor**:

- Takes no parameters.
- Automatically invoked to initialize members with default values.

2. **Parameterized Constructor**:
   - Takes one or more parameters.
   - Used to initialize objects with specific values.
3. **Characteristics**:
   - No return type.
   - Declared in the **public** section.

Code Example 1: Parameterized Constructor

```cpp
class Complex {
    int a, b;
public:
    Complex(int x, int y) { // Parameterized Constructor
        a = x;
        b = y;
    }
    void printNumber() {
        cout << "Your number is " << a << " + " << b << "i" << endl;
    }
};

```

Usage in Main:

```cpp
Complex c1(4, 6); // Implicit call
c1.printNumber();
Complex c2 = Complex(5, 7); // Explicit call
c2.printNumber();

```

Output:

```
Your number is 4 + 6i
Your number is 5 + 7i

```

Code Example 2: Points with Parameterized Constructor

```cpp
class Point {
    int x, y;
public:
    Point(int a, int b) { // Parameterized Constructor
        x = a;
        y = b;
    }
    void displayPoint() {
        cout << "The point is (" << x << ", " << y << ")" << endl;
    }
};

```

Usage in Main:

```cpp
Point p(1, 1);
```

```
p.displayPoint();
Point q(4, 6);
q.displayPoint();

```

Output:

```
The point is (1, 1)
The point is (4, 6)
```