Tutorial 35 - Destructor in C++

Key Concepts

1. **Destructor**:
    - Special member function automatically called when an object is destroyed.
    - Syntax: `~ClassName() { /* code */ }`
    - Does **not** take arguments or return values.
    - Used to release resources (e.g., memory, file handles) held by the object.
2. **Behavior**:
    - Called in the reverse order of object creation.
    - Destructor is invoked when:
        - Object goes out of scope.
        - Program execution ends.
        - `delete` is called for a dynamically allocated object.

Code Example and Explanation

Code Snippet 1: Destructor Example

```cpp
#include<iostream>
using namespace std;
int count = 0;
class num {
public:
    num() {
        count++;
        cout << "Constructor called for object number " << count << endl;
    }
    ~num() {
        cout << "Destructor called for object number " << count << endl;
        count--;
    }
};

```

Explanation:

1. **Global Variable**:
    - `count` keeps track of the number of active objects.
2. **Constructor**:
    - Increments `count` when an object is created and displays a message.
3. **Destructor**:
    - Decrements `count` when an object is destroyed and displays a message.

Code Snippet 2: Main Program

```cpp
int main() {
    cout << "Inside main function" << endl;
    cout << "Creating first object n1" << endl;
    num n1; // Constructor for n1 is called
    {
        cout << "Entering block" << endl;
        cout << "Creating two more objects" << endl;
        num n2, n3; // Constructors for n2 and n3 are called
        cout << "Exiting block" << endl;
    } // Destructors for n2 and n3 are called
    cout << "Back to main" << endl;
    return 0; // Destructor for n1 is called
}

```

Execution Flow:

1. `n1` is created, and its constructor is invoked.
2. Inside the block:
   - `n2` and `n3` are created, invoking their constructors.
   - At the end of the block, destructors for `n2` and `n3` are invoked in reverse order.
3. Exiting the main function, the destructor for `n1` is invoked.

Output

```
Inside main function
Creating first object n1
Constructor called for object number 1
Entering block
Creating two more objects
Constructor called for object number 2
Constructor called for object number 3
Exiting block
Destructor called for object number 3
Destructor called for object number 2
Back to main
Destructor called for object number 1

```

Short Notes for Notebook

Destructor in C++

1. **Definition**:
   - Special member function automatically invoked when an object is destroyed.
   - Syntax: `~ClassName() { /* code */ }`
2. **Characteristics**:
   - No arguments, no return type.
   - Automatically called when:
     - Object goes out of scope.
     - Program ends.
     - Dynamically allocated object is deleted.
   - Used for cleanup (e.g., releasing memory, closing files).
3. **Code Example**:

```
class num {
    static int count;
public:
    num() { count++; cout << "Constructor called for " << count << endl; }
    ~num() { cout << "Destructor called for " << count << endl; count--; }
};

```

4. **Behavior**:
   - **Creation**: Constructor runs when objects are created.
   - **Destruction**: Destructor runs in reverse order when objects go out of scope.

Execution Flow Example

```
int main() {
    num n1;         // Constructor for n1
    {
        num n2, n3; // Constructors for n2, n3
    }               // Destructors for n3, n2
    return 0;       // Destructor for n1
}

```

Output:

```
Constructor called for 1
Constructor called for 2
Constructor called for 3
Destructor called for 3
Destructor called for 2
Destructor called for 1
```