C Enumeration

Topics : C
Written on April 13, 2023

In C programming language, an enumeration is a user-defined data type that consists of a set of named constants, also known as enumerators. Enumerations are defined using the `enum` keyword, which defines a new type that can be used to declare variables of that type.

Here's the syntax for defining an enumeration:

```
enum <enumeration_name> {
    <enumerator_1>,
    <enumerator_2>,
    // ...
};
```

For example, let's define a `Color` enumeration that represents different colors:

```
enum Color {
    RED,
    GREEN,
    BLUE
};
```

Once an enumeration is defined, you can declare variables of that type:

```
enum Color c1;
```

You can also initialize an enumeration variable at the time of declaration:

```
enum Color c2 = GREEN;
```

You can access individual enumerators of an enumeration using their names:

```
c1 = RED;
printf("c1 = %d\n", c1); // Output: c1 = 0
printf("c2 = %d\n", c2); // Output: c2 = 1
```

In C, the first enumerator in an enumeration has a default value of 0, and subsequent enumerators

are assigned values that are one greater than the previous enumerator. However, you can explicitly assign values to individual enumerators:

```
enum Color {
    RED = 1,
    GREEN = 2,
    BLUE = 4
};
```

Here's an example program that demonstrates the use of enumerations in C:

```
#include <stdio.h>

enum Color {
    RED,
    GREEN,
    BLUE
};

int main() {
    enum Color c1;
    enum Color c2 = GREEN;

    c1 = RED;
    printf("c1 = %d\n", c1); // Output: c1 = 0
    printf("c2 = %d\n", c2); // Output: c2 = 1

    return 0;
}
```

This program declares a `Color` enumeration, declares two variables `c1` and `c2`, initializes `c2` with the value `GREEN`, updates `c1` with the value `RED`, and prints out the values of both variables.