ANSWER:1

**The Role of Format Specifiers in C Programming**

In C programming, the process of printing information plays a crucial role. Format specifiers determine how data is displayed. They instruct the compiler on how to interpret and present data types, such as integers, floating-point numbers, or characters.

**Using the printf Function**

To display formatted output in C, the printf() function is employed. It specifies how variable values should be read and displayed accurately. Each format specifier corresponds to a particular data type.

**Key Roles of Format Specifiers**

1. Identify Data Type:

They guide the printf function on how to interpret a variable's data type, such as integer, floating-point, character, or string.

2. Control Appearance:

They allow customization of the output format, such as width, precision, and padding for displaying values.

3. Ensure Type Safety:

When used correctly, they ensure that the function processes the correct data type. A mismatch can produce incorrect results.

**Common Format Specifiers**

1. Signed Integer: %d or %i

int x = 5;

printf("%d", x); // Output: 5

2. Unsigned Integer: %u

unsigned int a = 43;

printf("%u", a); // Output: 43

3. Floating-point Number: %f

float i = 3.123456;

printf("%f", i); // Output: 3.123456

4. Double Precision Floating-point: %lf

double pi = 6753.14159265358979;

printf("%lf", pi); // Output: 6753.141593

5. Single Character: %c

char ch = 'd';

printf("%c", ch); // Output: d

6. String: %s

char str[] = "C Programming";

printf("%s", str); // Output: C Programming

7. Hexadecimal (Lowercase): %x

int a = 255;

printf("%x", a); // Output: ff

8. Hexadecimal (Uppercase): %X

int num = 255;

printf("%X", num); // Output: FF

9. Octal Format: %o

int num = 8;

printf("%o", num); // Output: 10

10. Percentage Symbol: %%

printf("This is 100%% correct!"); // Output: This is 100% correct!

11. Pointer Address: %p

int x = 10;

int \*ptr = &x;

printf("%p", ptr); // Output: (Memory address of x)

12. Scientific Notation: %e or %E

double a = 3.14159;

printf("%e", a); // Output: 3.141590e+00