Storage Class Example: Used to pass data to functions

- Retains value between function calls

Syntax:

```
static int count = 0;
```

7.5 Scope Rules

Scope determines where a variable is accessible.

- **Types of Scope:**
  - Local scope
  - Global scope
  - Block scope

Syntax:

```
int x = 10;  // global
```

7.6 Category of Functions

7.6.1 No Arguments, No Return Value:

Syntax:

```
void display()
{
    printf("Hello");
}
```

7.6.2 Arguments, No Return Value

Syntax:

```
void sum(int a, int b)
{
    printf("%d", a + b);
}
```

7.6 Category of Functions

7.6.3 Arguments and Return Value

Syntax:

```
int sum(int a, int b)
{
    return a + b;
}
```

7.6.4 No Arguments, Return Value

Syntax:

```
int getNum()
{
    return 10;
}
```

7.7 Recursive Functions

- Function calling itself
- Must have base condition

Example:

```
int fact(int n)
{
    if(n == 0)
        return 1;
    return n * fact(n - 1);
}
```

7.8 Call by Value

- Copy of value is passed
- Original value unchanged

Syntax:

```
void change(int x)
{
    x = 20;
}
```

7.8 Call by Reference

- Address is passed
- Original value changes

Example:

```
void change(int *x)
{
    *x = 20;
}
```

7.9 Passing Array to Function

- Arrays are passed by reference

Syntax:

```
void display(int a[], int n)
{
    for(int i = 0; i < n; i++)
        printf("%d ", a[i]);
}
```

Passing String to Function

Example:

```
void show(char str[])
{
    printf("%s", str);
}
```

Common Errors

- Missing function prototype
- Incorrect return type
- Argument mismatch
- Infinite recursion

Best Practices

- Use meaningful function names
- Keep functions small
- Avoid global variables
- Always declare functions before use