1. `break` Statement

The `break` statement is used to terminate a loop or `switch` statement immediately.

Example: Using `break` in a Loop

```
#include <stdio.h>

int main() {
    printf("Example of break:\n");
    for (int i = 1; i <= 10; i++) {
        if (i == 5) {
            break; // Exit the loop when i equals 5
        }
        printf("%d ", i); // Print numbers before breaking
    }
    printf("\nExited the loop.\n");
    return 0;
}
```

Output:

```
Example of break:
1 2 3 4
Exited the loop.
```

---

2. `continue` Statement

The `continue` statement skips the remaining part of the loop's current iteration and moves to the next iteration.

Example: Using `continue` in a Loop

```
#include <stdio.h>

int main() {
    printf("Example of continue:\n");
    for (int i = 1; i <= 10; i++) {
        if (i % 2 == 0) {
            continue; // Skip even numbers
        }
        printf("%d ", i); // Print only odd numbers
    }
    printf("\nSkipped even numbers.\n");
    return 0;
}
```

Output:

```
Example of continue:
1 3 5 7 9
Skipped even numbers.
```

---

3. `goto` Statement

The `goto` statement transfers control to a labeled statement elsewhere in the program. It can be useful for breaking out of deeply nested loops or handling errors.

Example: Using `goto` to Exit a Nested Loop

```
#include <stdio.h>

int main() {
    printf("Example of goto:\n");
    for (int i = 1; i <= 3; i++) {
        for (int j = 1; j <= 3; j++) {
            if (i == 2 && j == 2) {
                goto exit_loop; // Exit both loops when i = 2 and j = 2
            }
            printf("i = %d, j = %d\n", i, j);
        }
    }

exit_loop:
    printf("Exited nested loops using goto.\n");
    return 0;
}
```

Output:

```
Example of goto:
i = 1, j = 1
i = 1, j = 2
i = 1, j = 3
i = 2, j = 1
Exited nested loops using goto.
```

- **`break`**: Ends a loop or a `switch` statement immediately.
- **`continue`**: Skips the rest of the current iteration and proceeds to the next iteration of the loop.
- **`goto`**: Directs the program flow to a labeled statement. Use with caution to avoid unstructured code.