**Faculty of Information Technology, University of Moratuwa**
**BSc. (Hons) in Information Technology**
**BSc. (Hons) in Artificial Intelligence**
**IN1101 - Programming Fundamentals**

**Level 1 – Semester 1**

---

# Jump Statements

C has four statements that perform an unconditional branch: **return, goto, break** and **continue**. Of these, you can use return and goto inside a function. You can use the break and continue statements in conjunction with any loop statements.

## The return Statement

The return statement is used to return from a function. We will discuss the function later. It is categorised as a jump statement because it causes execution to return (jump back) to the point at which the call to the function was made. A return may or may not have a value associated with it. A return with a value can be used only in a function with a non-void return type. In this case, the value associated with the return becomes the return value of the function. A return without a value is used to return from a void function.

## The goto Statement

The goto statement in C has a label as its object. When the goto statement executes, control transfers to the C statement that follows the label where it is defined. The label definition must reside in the function that references it and must be defined only once. A label name follows the standard naming rules of C and has a colon attached to its right side. When a goto statement refers to a label, the colon is not used.

The general form of the goto statement is

goto label;

.

.

label:

Where the label is any valid label either before or after goto. For example, you could create a loop from 1 to 7 using the goto and a label, as shown here:

```
#include<stdio.h>
void main() {
    int x = 1;
    loop1:
    printf("%d", x);
    x++;
    if (x <= 7)
        goto loop1;
}
```

**The break Statement**

The break statement has two uses. You can use it to terminate a case in the switch statement. You can also use it to force immediate termination of a loop, bypassing the normal loop conditional test. The syntax is as follows:

break;

When the break statement is encountered inside a loop, the loop is immediately terminated, and program control resumes at the next statement following the loop.

For example,

```
#include<stdio.h>
int main() {
    int t;
    for (t = 0; t < 100; t++) {
        printf("%d", t);
        if (t == 10)
            break;
    }
    return 0;
}
```

Output: 12345678910

Prints the numbers 0 through 10 on the screen. Then the loop terminates because the break causes an immediate exit from the loop, overriding the conditional test t<100.

**The continue Statement**

The continue statement works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between. For the for loop, continue causes the increment and then the conditional test portions of the loop to execute. For the while and do while loops, program control passes to the conditional tests.

The syntax is:

continue Example:

```
for (int i = 0; i < 10; i++) {
    if (c == 'y')
        break;
    else
        continue;
}
```