Control Statements:Conditional Statements

- Introduction, conditional execution (if, if-else, nested if), and selection (switch), unconditional types (break, continue, goto).

Unconditional Jump Statements-(break, continue, goto, exit)

The break statement:

- The break statement is an unconditional jump statement.
- The break statement can be used as the last statement in each case's statement list in switch.
- A break statement causes control to transfer to the end of the switch statement.
- If a break statement is not used, the flow of control will continue into the next case.

"What is a use of break statement?"

The break statement in C programming language has the following two usages:

- 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.
- It can be used to terminate a case in the switch statement.
- If you are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Flow without *break*	**Flow with*break***

switch (option)
{
case 'A':
aCount++;
case 'B':
bCount++;
case 'C':
cCount++;
}
	switch (option)
	{
	case 'A':
	aCount++;
	break;
	case 'B':
	bCount++;
	break;
	case 'C':
	cCount++;
	break;
	}

Common Programming Error:

- Not including break statement as the last statement in a case block will also result in execution of the next case block. This will give undesired results.

The continue Statement

- Like the break statement, the continue statement can be placed only in the body of a for loop, or a while loop, or a do...while loop.
- When a continue statement executes, control is sent to the beginning of the loop.
- The statements between the continue statement and the end of the loop aren't executed.

Flow without *continue*	**Flow with*continue***

for(exp1;exp2;exp3)
{
Action 1;
---------- 	
---------- 	
Action N;
}
	for(exp1;exp2;exp3)
	{
	Action 1;
	continue;
	---------- 	
	Action N;

#include<stdio.h>
void main()
{
int i;
for(i=1;i<=3;i++)
{
printf("INDIA");
printf("is our country\n");
}
getch()
}
 	
OUTPUT:
INDIA is our country
INDIA is our country
INDIA is our country
	#include<stdio.h>
	void main()
	{
	int i;
	for(i=1;i<=3;i++)
	{
	printf("INDIA\n");
	continue;
	printf("is our country\n");
	}
	getch()
	}
	
	OUTPUT:
	INDIA
	INDIA
	INDIA

The goto Statement

- The goto statement is used to transfer the control of the program from one point to another.
- It is something referred to as unconditionally branching.
- The goto is used in the form

goto label;

- **Label statement:** The label is a valid 'C' identifier followed by a colon. We can precode any statement by a label in the form

Label :statement ;

- This statement immediately transfers execution to the statement labelled with the label identifier.

Example:

```
#include<stdio.h>
void main()

{

  int i=1;

  Back:  printf("INDIA\n");

         if(i!=5)
           {
            i++;
            goto back;
           }
}
```

OUTPUT:
INDIA
INDIA
INDIA
INDIA
INDIA

The exit() statement:

- The function exit() will terminate the process/program that calls the exit.

Note:

void exit (int status);

- On call the process will terminate normally. It will perform regular cleanup as normal for a normal ending process.
- Parameters: **A status value returned to the parent process.**
- **Return value:** The argument status is returned to the host environment. Normally you say 1 or higher if something went wrong and 0 if everything went ok.
- **"What is the difference between exit(0) and exit(1)?"**
  - ✓ **exit(0):** indicates successful program termination & it is fully portable.
  - ✓ **exit(1):** indicates unsuccessful termination. However, it's usage is non-portable.
- **"What are the differences between continue and break statement?"**
  - ✓ A break statement results in the termination of the statement to which it applies (switch, for, do, or while).
  - ✓ A continue statement is used to end the current loop iteration and return control to the loop statement / start.