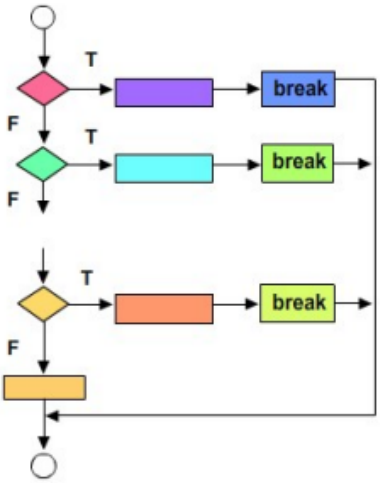
NOTES

Bachelor of Science (First Year)
CSC-101-Programming fundamentals using C
Title of the Unit : V – Overview of C
Module Name : Control constructs: branching
Module Number : 23

Switch Statement

C's most flexible program control statement is the switch statement, which lets the program execute different statements based on an expression that can have more than two values.Earlier control statements, such as if, were limited to evaluating an expression that could have only two values: True or false. To control program flow based on more than two values, the user had to use multiple nested if statements. Its purpose is to allow the value of a variable or expression to control the flow of program execution. A switch statement is defined across many individual statements.




```
switch(expression)
{
case constant1:
statements 1;
break;
case constant2:
statements 2;
break;
.....
default:
statements n;
break;
}
```

From the statement, expression is any expression that evaluates to an integer value: type long, int, or char. The switch statement evaluates expression and compares the value against the templates following each case label, and then one of the following happens:

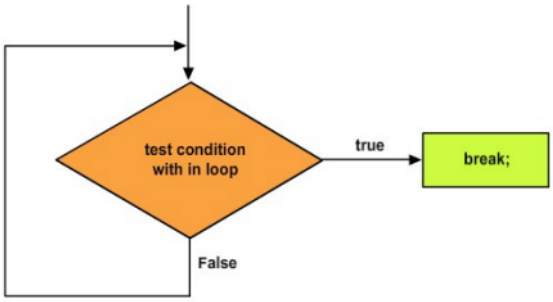
If a match is found between expression and one of the templates, execution is transferred to the statement that follows the case label. If no match is found, execution is transferred to the statement following the optional default label. If no match is found and there is no default label, execution passes to the first statement following the switch statement's closing brace.

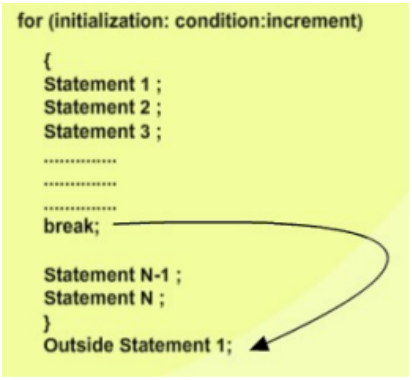
```
int main()
{
    int num=2;
    switch(num+2)
    {
       case 1:
         printf("Case1: Value is: %d", num);
       case 2:
         printf("Case1: Value is: %d", num);
       case 3:
         printf("Case1: Value is: %d", num);
       default:
         printf("Default: Value is: %d", num);
    }
    return 0;
}
```

When provided in the switch should result in a constant value otherwise it would not be valid. Duplicate case values are not allowed.The default statement is optional.Even if the switch case statement do not have a default statement, it would run without any problem.The break statement is used inside the switch to terminate a statement sequence. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.Switch statements are limited to **integer** values only in the check condition.

Break Statement

The break statement can be placed in a switch case to exit the switch case once a matching case if found.Break statement can also be used in the body of a ‘for’ loop, ‘while’ loop, or ‘do...while’ loopWhen a break statement is encountered, execution exits the loop.Generally a break statement is preceded with an if condition inside the body of the loop





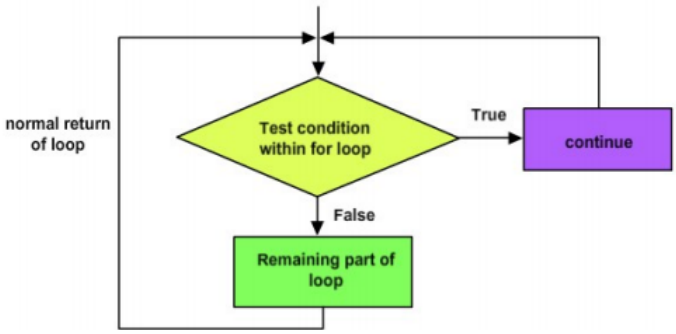
Break Statement in switch case

The break statement is used to terminate a case in the switch statement after a matching case is found.When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.The break statement is optional in switch case. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.The first iteration of for loop will print value of i as 1. At the second iteration, i=2 and the loop breaks here The while loop starts with value of i as 2, prints 2 and increments i to 3, checks condition ,prints 3, increments i to 4. When I is 4, the loop exits due to the break.The do while loop starts with value of I as 4. The value 4 is printed, then I incremented to 5, condition checked and value 5 printed andi is incremented to 6, condition checked and since I is 6, the do while loop ends with the break.When a break statement is encountered inside a nested loop, it causes the program to exit the innermost loop only

```
int main()
{
int i;
clrscr();
for(i=1;i<=10;i=i+1)
{
if(i==2)
break;
printf("%d\n",i);
}
while(i<=10)
{
if(i==4)
break;
printf("%d\n",i);
i=i+1;
}
d0
{
if(i==6)
break;
printf("%d\n",i);
i=i+1;
}
while(i<=10)
getch();
return 0;
}
```

Continue Statement

Like the break statement, the continue statement can be placed only in the body of a ‘for’ loop, a while loop, or a ‘do...while’ loop.When a continue statement executes, the next iteration of the enclosing loop begins immediately. The statements between the continue statement and the end of the loop aren′t executed Generally, Continue is used inside a loop preceded with an if condition. It causes the control of a program to skip the rest of the current iteration of a loop and start the next iteration.



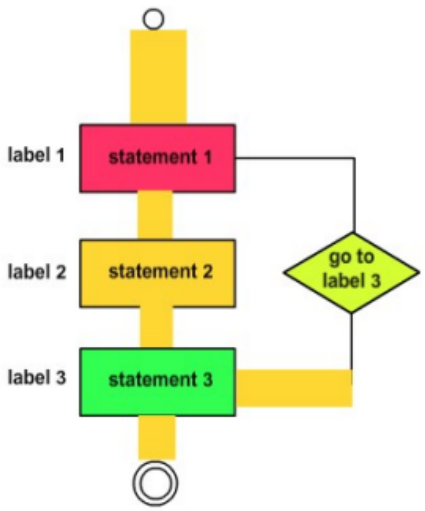

```
..
for (int j=0; j<=8; j++)
{
   if (j==4)
   {
       continue;
   }

   printf("%d ", j);
}
..
```

The for loop continues exeution from j=0 to j=3 and prints value of j from 0 to 3 every time in the body of the loop.When j become 4, we encounter a continue satement inside the body of the loop. The continue statement skips the printf() statement, ends body of the loop once and takes the control to the increment statement j++ which makes j to 5.The loop then continues normally from j=5 to 8 printing the valuses of j from 5 to 8 and ends when j becomes 9.Thus the program printf the output

0 1 2 3 5 6 7 8

Go to Statement

The goto statement is one of C′s unconditional jump, or branching statements When program execution reaches a goto statement, execution immediately jumps, or branches, to the location specified by the goto statement. This statement is unconditional because execution always branches when a goto statement is encountered; the branch doesn′t depend on any program conditions. A goto statement and its target must be in the same function, but they can be in different blocks. Often, a break statement, a continue statement, or a function call can eliminate the need for a goto statement.



The use of goto statement is highly discouraged as it makes the program logic very complex.use of goto makes the task of analyzing and verifying the correctness of programs (particularly those involving loops) very difficult.Use of goto can be simply avoided using break and continue statements.