C Language (Lecture 5)

Break and Continue statements:

There are two statements built in C programming, break; and continue; to alter the normal flow of a program. Loops perform a set of repetitive task until text expression becomes false but it is sometimes desirable to skip some statement/s inside loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used. The break; statement is also used in switch statement to exit switch statement.

break Statement

In C programming, break is used in terminating the loop immediately after it is encountered. The break statement is used with conditional if statement.

Syntax of break statement

```
break;
```

The break statement can be used in terminating all three loops for, while and do...while loops.

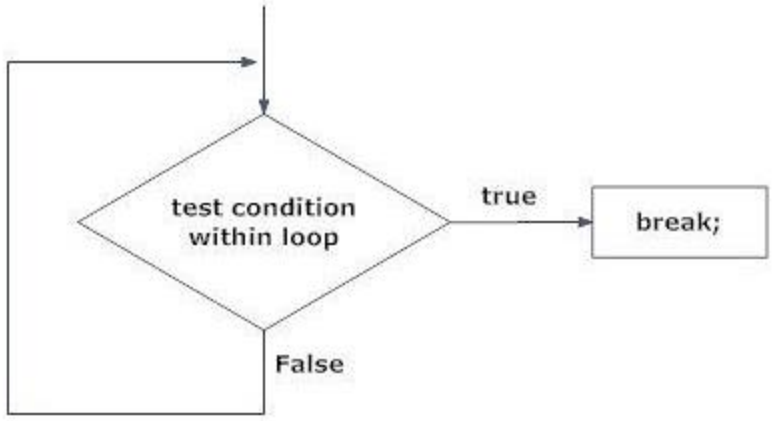

Fig : Flowchart of break statement

The figure below explains the working of break statement in all three type of loops.

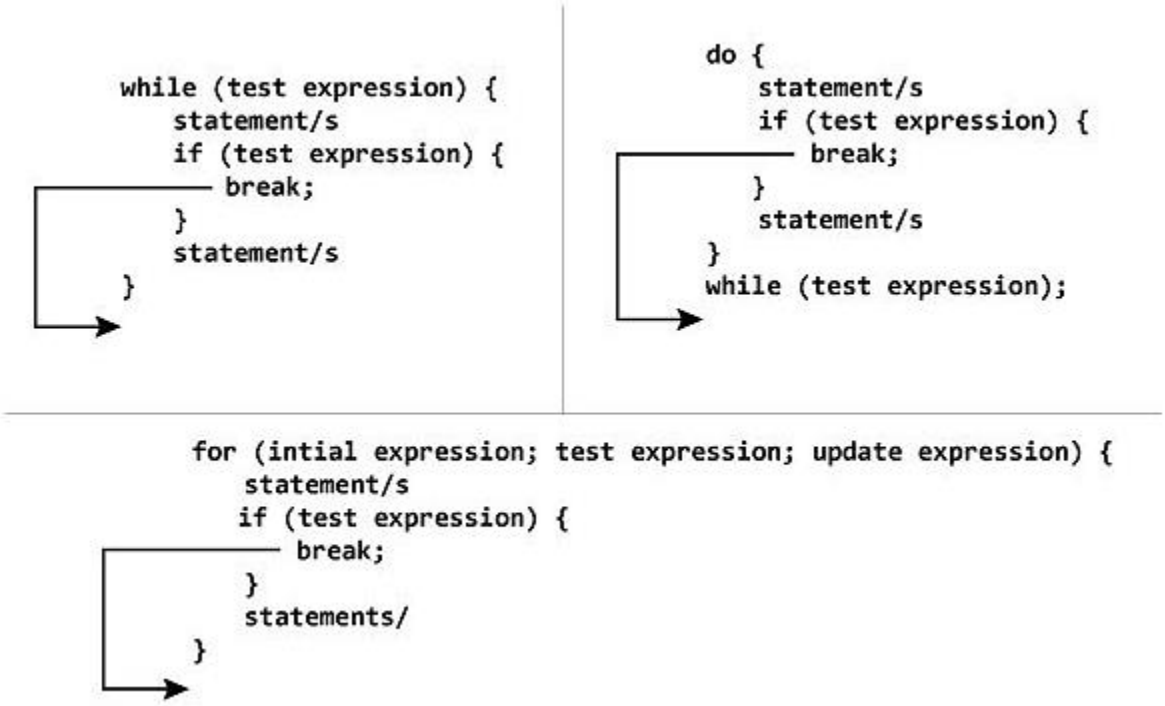

Fig : Use of break statement in different types of loops

Example of break statement

C program to find average of maximum of ***n*** positive numbers entered by user. But, if the input is negative, display the average(excluding the average of negative input) and end the program.

```
/* C program to demonstrate the working of break statement by
terminating a loop, if user inputs negative number*/
# include <stdio.h>
int main(){
   float num,average,sum;
   int i,n;
   printf("Maximum no. of inputs\n");
   scanf("%d",&n);
   for(i=1;i<=n;++i){
       printf("Enter n%d: ",i);
       scanf("%f",&num);
       if(num<0.0)
       break;                    //for loop breaks if num<0.0
       sum=sum+num;
```

```
}
  average=sum/(i-1);
  printf("Average=%.2f",average);
  return 0;
}
```

Output:

```
Maximum no. of inputs
4
Enter n1: 1.5
Enter n2: 12.5
Enter n3: 7.2
Enter n4: -1
Average=7.07
```

In this program, when the user inputs number less than zero, the loop is terminated using break statement with executing the statement below it i.e., without executing sum=sum+num.

continue Statement

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statements are used.

Syntax of continue Statement

```
continue;
```

Just like break, continue is also used with conditional if statement.

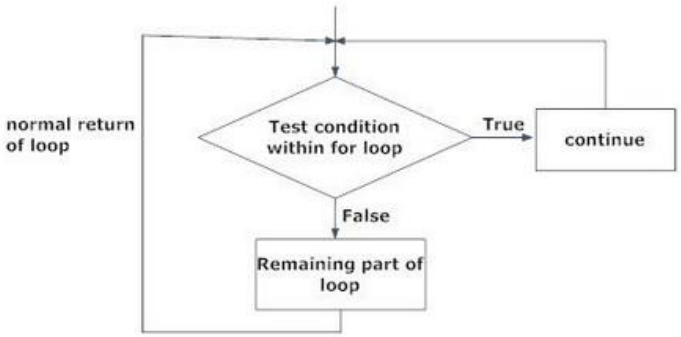

Fig 2.9 Flowchart of continue statement

For better understanding of how continue statements works in C programming. Analyze the figure below which bypasses some code/s inside loops using continue statement.

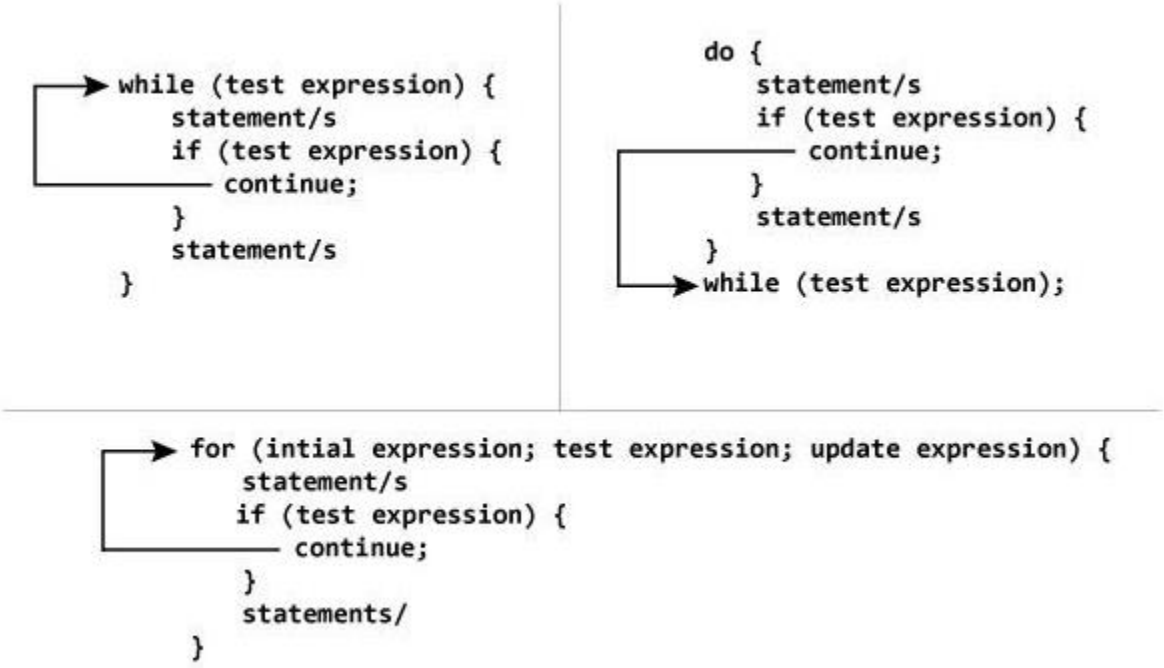

Fig 2.10 Use of continue statement

Example of continue statement

C program to find the product of 4 integers entered by a user. If user enters 0 skip it.

```
//program to demonstrate the working of continue statement in C programming
# include <stdio.h>
int main(){
    int i,num,product;
    for(i=1,product=1;i<=4;++i){
        printf("Enter num%d:",i);
        scanf("%d",&num);
        if(num==0)
            continue;  / *In this program, when num equals to zero, it skips
the statement product*=num and continue the loop. */
        product*=num;
}
    printf("product=%d",product);
return 0;
}
```

Output:

```
Enter num1:3
Enter num2:0
Enter num3:-5
Enter num4:2
product=-30
```

switch—case statement

Decision making are needed when, the program encounters the situation to choose a particular statement among many statements. If a programmer has to choose one block of statement among many alternatives, nested if...else can be used but, this makes programming logic complex. This type of problem can be handled in C programming using switch statement.

Syntax of switch...case

```
switch (n) {
case constant1:
   code/s to be executed if n equals to constant1;
   break;
case constant2:
   code/s to be executed if n equals to constant2;
   break;
   .
   .
   .
default:
   code/s to be executed if n doesn't match to any cases;
}
```

The value of *n* is either an integer or a character in above syntax. If the value of *n* matches constant in case, the relevant codes are executed and control moves out of the switch statement. If the *n* doesn't matches any of the constant in case, then the default codes are executed and control moves out of switch statement.

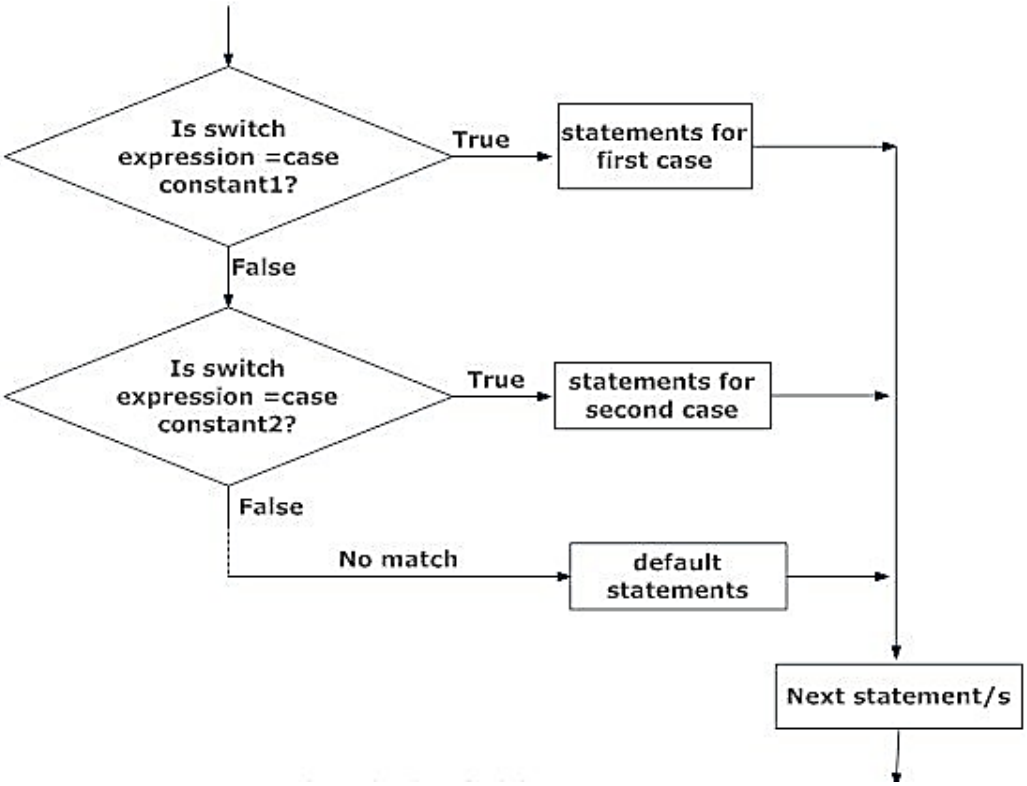

Fig : Flowchart of switch-case statement

Example of switch...case statement

C program that asks user an arithmetic operator('+','-','*' or '/') and two operands and perform the corresponding calculation on the operands.

```
/* C program to demonstrate the working of switch...case
statement */
/* C Program to create a simple calculator for addition,
subtraction,
   multiplication and division */

# include <stdio.h>
int main() {
    char o;
    float num1,num2;
    printf("Select an operator either + or - or * or / \n");
    scanf("%c",&o);
    printf("Enter two operands: ");
    scanf("%f%f",&num1,&num2);
```

```
    switch(o) {
        case '+':
            printf("%.1f + %.1f = %.1f",num1, num2, num1+num2);
            break;
        case '-':
            printf("%.1f - %.1f = %.1f",num1, num2, num1-num2);
            break;
        case '*':
            printf("%.1f * %.1f = %.1f",num1, num2, num1*num2);
            break;
        case '/':
            printf("%.1f / %.1f = %.1f",num1, num2, num1/num2);
            break;
        default:
            /* If operator is other than +, -, * or /, error
message is shown */
            printf("Error! operator is not correct");
            break;
    }
    return 0;
}
```

Output:

```
Enter operator either + or - or * or /
*
Enter two operands: 2.3
4.5
2.3 * 4.5 = 10.3
```

The break statement at the end of each case cause switch statement to exit. If break statement is not used, all statements below that case statement are also executed.

goto statement

In C programming, goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

Syntax of goto statement

```
goto label;
.............
.............
.............
label:
statement;
```

In this syntax, label is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the label: and executes the code below it.

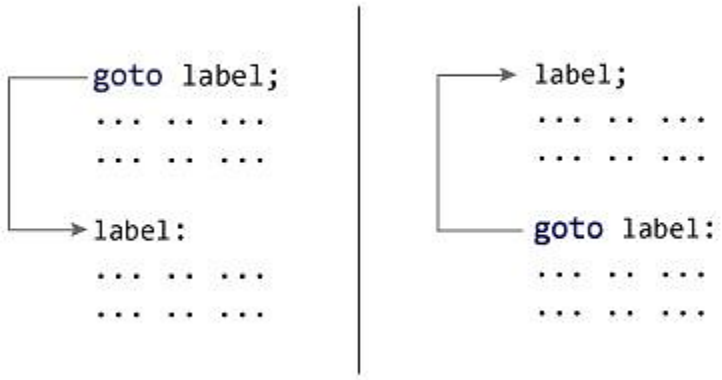

Fig: Flowchart of goto statement

Example of goto statement

```
/*  C program to demonstrate the working of goto statement. */
/* This program calculates the average of numbers entered by
user. */
/* If user enters negative number, it ignores that number and
   calculates the average of number entered before it.*/

# include <stdio.h>
int main(){
```

```
    float num,average,sum;
    int i,n;
    printf("Maximum no. of inputs: ");
    scanf("%d",&n);
    for(i=1;i<=n;++i){
        printf("Enter n%d: ",i);
        scanf("%f",&num);
        if(num<0.0)
        goto jump;             /* control of the program moves to
label jump */
        sum=sum+num;
}
jump:
  average=sum/(i-1);
  printf("Average: %.2f",average);
  return 0;
}
```

Output:

```
Maximum no. of inputs: 4
Enter n1: 1.5
Enter n2: 12.5
Enter n3: 7.2
Enter n4: -1
Average: 7.07
```

Though goto statement is included in ANSI standard of C, use of goto statement should be reduced as much as possible in a program.

Reasons to avoid goto statement

Though, using goto statement give power to jump to any part of program, using goto statement makes the logic of the program complex and tangled. In modern programming, goto statement is considered a harmful construct and a bad programming practice. The goto statement can be replaced in most of C program with the use of break and continue statements. In fact, any program in C programming can be perfectly written without the use of goto statement. All programmer should try to avoid goto statement as possible as they can.

Summary:

In this lecture we studied about if, if-else and nested if-else structure with illustrating examples. Concept of for , while and do-while loops are also introduced with various examples .Finally use of break, continue, switch-case and goto is discussed for the programming purpose.