Parul® University

6. Control Structure in C

Sushil Kumar, Assistant Professor
Computer Science & Engineering

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-6

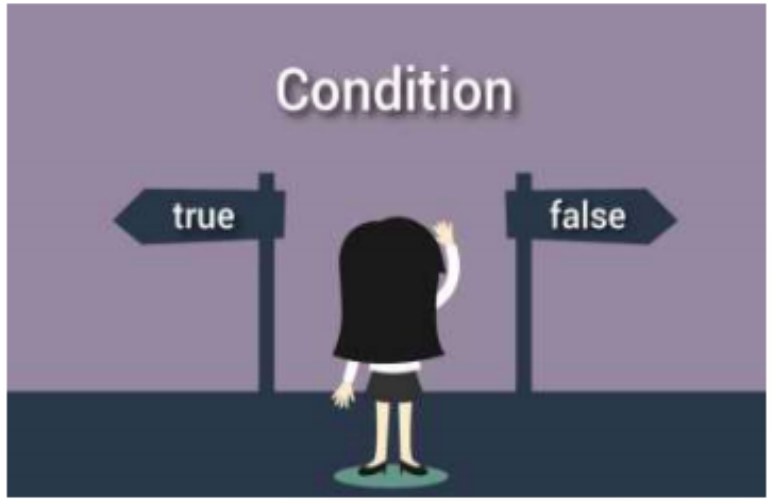
Control Structure in C

Contents

1. Decision Making

- **Decision making** is used to specify the order in which statements are executed.

- **Decision making in a C program using:-**
  - **if statement**
  - **if…else statement**
  - **if…else if…else statement**
  - **nested if...else statement**
  - **Switch case Statement**



1.1 if statement

```
if (testExpression)
{
   // statements
}
```

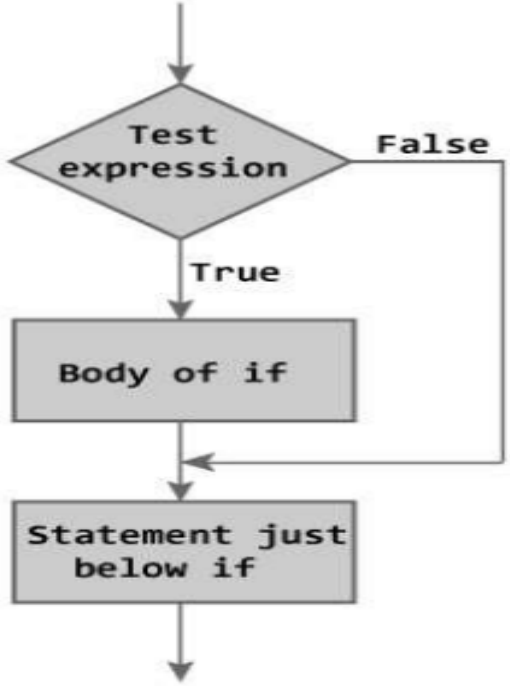


Figure: Flowchart of if Statement

Example: if statement

- **Program to display a number if user enters negative number If user enters positive number, that number won't be displayed.**

```
#include <stdio.h>
int main()
{
int number;
printf("Enter an integer: ");
scanf("%d", &number);
// Test expression is true if number is less than 0
if (number < 0) {
printf("You entered %d.\n", number);
}
printf("The if statement is easy.");
return 0;
}
```

1.2 if...else statement

- The if...else statement executes some code if the test expression is true (nonzero) and some other code if the test expression is false (0).

Syntax of if...else

```
if (test Expression)
{
        // codes inside the body of if
}
else
{
        // codes inside the body of else
}
```

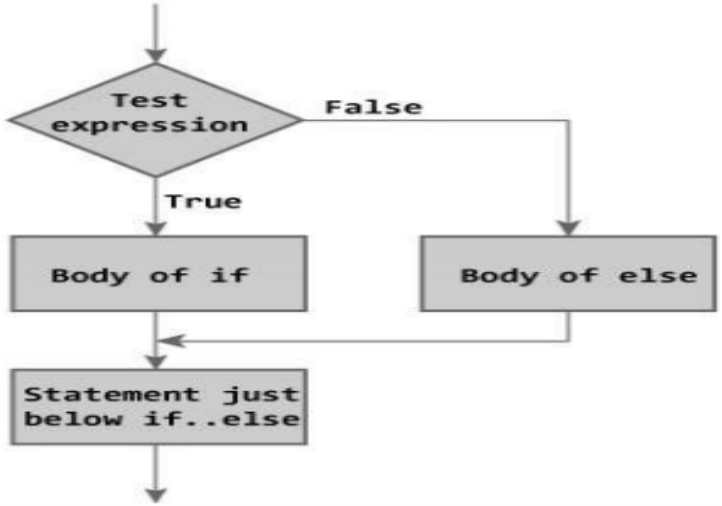


Figure: Flowchart of if...else Statement

Example: if...else statement

```
// Program to check whether an integer entered by the user is odd or even
#include <stdio.h>
int main()
{
        int number;
        printf("Enter an integer: ");
        scanf("%d",&number);
        // True if remainder is 0
        if( number%2 == 0 )
        printf("%d is an even integer.",number);
        else
        printf("%d is an odd integer.",number);
        return 0;
}
```

1.3 if...else if....else Statement

- The if...else statement executes two different codes depending upon whether the test expression is true or false. Sometimes, a choice has to be made from more than 2 possibilities.

- The if...else if…else statement allows you to check for multiple test expressions and execute different codes for more than two conditions

Syntax of if...else if....else statement.

```
if (testExpression1) {
   // statements to be executed if testExpression1 is true
} else if(testExpression2) {
   // statements to be executed if testExpression1 is false and
testExpression2 is true
} else if (testExpression 3) {
   // statements to be executed if testExpression1 and
testExpression2 is false and testExpression3 is true
} else {
   // statements to be executed if all test expressions are false
}
```

Example: if...else if....else statement

```
// Program to relate two integers
using =, > or <
#include <stdio.h>
int main()
{
int number1, number2;
printf("Enter two integers: ");
scanf("%d %d", &number1,&number2);
//checks if two integers are equal.
if(number1 == number2)
{
        printf("Result: %d = %d“,number1,number2);
}
```

```
//checks if number1 is greater than number2.
else if (number1 > number2)
{
printf("Result: %d > %d",
number1, number2);
}
// if both test expression is false
else {
printf("Result: %d < %d",number1, number2);
}
return 0;
}
```

1.4 Nested if else statement

- **Nested if else statement** is same like **if else statement**, where new block of if else statement is defined in existing if or else block statement.

- Used when user want to check **more than one conditions at a time.**

Syntax of Nested If else Statement

```
if(condition is true){
    if(condition is true){
        statement;
    }else{
        statement;
    }
}else{
    statement;
}
```

```
if (test condition - 1)
  {
    if (test condition - 2)
      {
        statement 1;
      }
    else
      {
        statement 2;
      }
  }
else
  {
    statement 3;
  }
statement x;
```

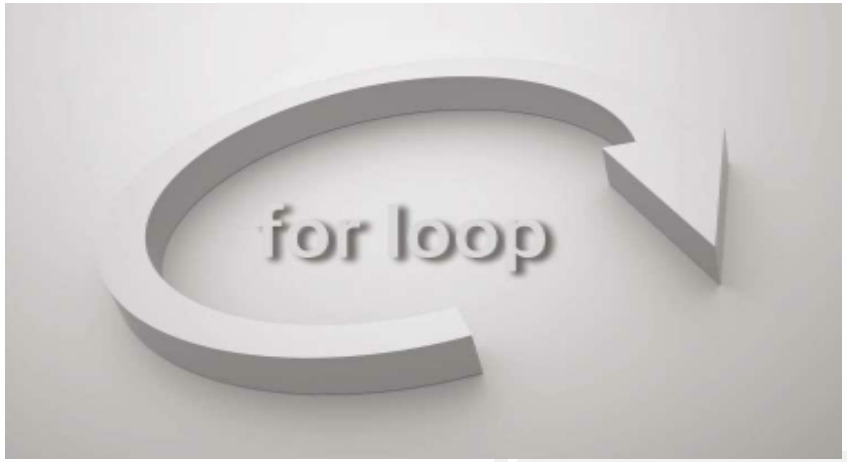
Example of Nested if else Statement

```
#include <stdio.h>
void main(){
        char username;
        int password;

        printf("Username:");
        scanf("%c",&username);
        printf("Password:");
        scanf("%d",&password);

        if(username=='a'){
                if(password==12345){
                        printf("Login successful");
                }else{
                        printf("Password is incorrect, Try again.");
                }
        }else{
                printf("Username is incorrect, Try again.");
        }
}
```

2. Loops

- **Loops** are used in programming to repeat a specific block until some end condition is met.

- **There are three loops in C programming:**
  - for loop
  - while loop
  - do...while loop
  - Nested loops



2.1 for Loop

- **The syntax of a for loop is:**

```
for (initialization Statement; testExpression; update Statement)
{
    // codes
}
```

for loop Flowchart

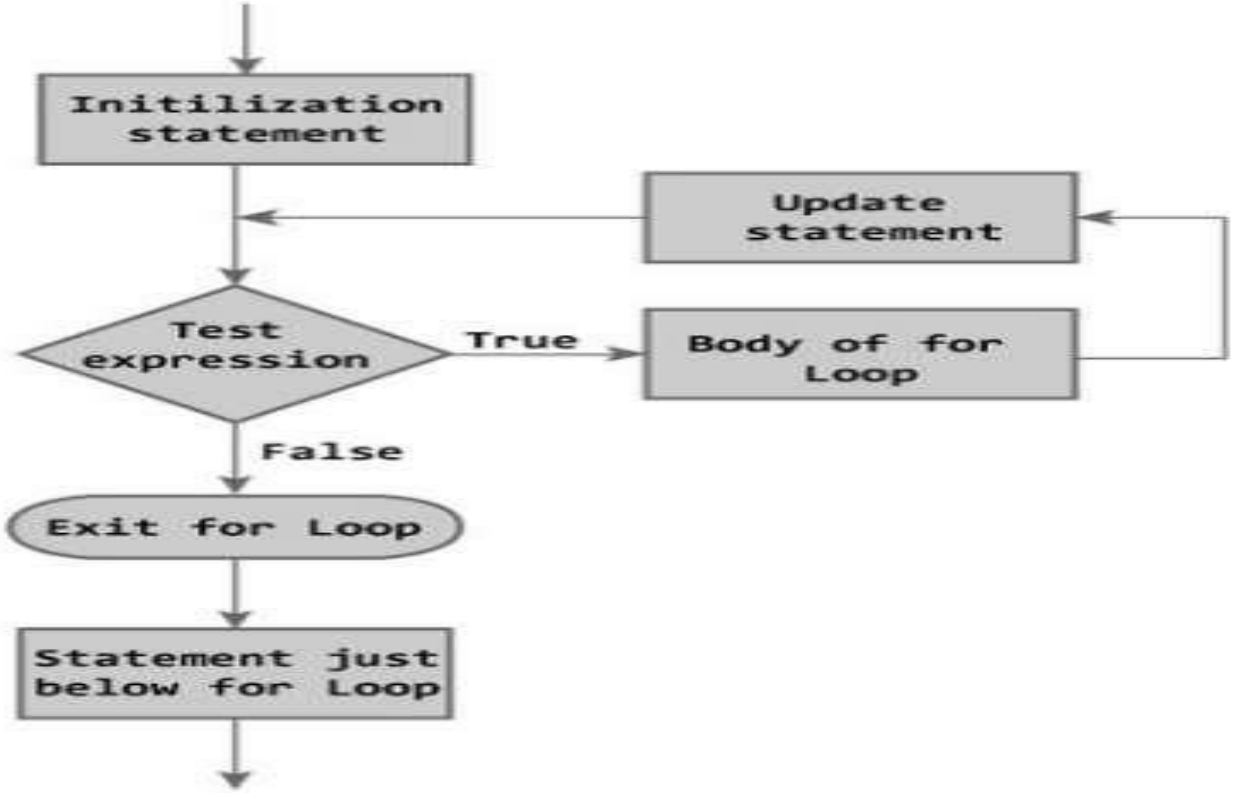


Figure: Flowchart of for Loop

Example: for loop

```
// Program to calculate the sum of
first n natural numbers

// Positive integers 1,2,3...n are  known as
natural numbers

#include <stdio.h>

int main(){

    int n, count, sum = 0;

    printf("Enter a positive integer: ");

    scanf("%d", &n);
```

```
for(count = 1; count <= n; ++count)
    {
        sum += count;
    }
    printf("Sum = %d", sum);  return 0;

}
```

2.2 while loop

- The syntax of a while loop is:

```
while (testExpression)
{
   //codes
}
```

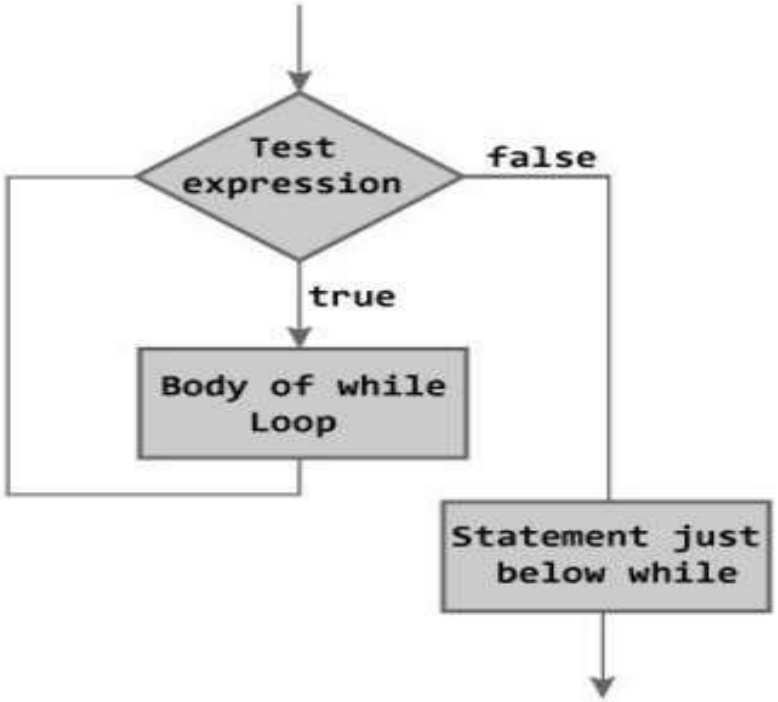


Figure: Flowchart of while Loop

Example: while loop

```
/ Program to find factorial of
a number

// For a positive integer n,  factorial = 1*2*3...n


#include <stdio.h>

int main(){
    int number;  long factorial;

    printf("Enter an integer: ");  scanf("%d",&number);

    factorial = 1;
```

```
// loop terminates when number is less  than or
  equal to 0

  while (number > 0) {
    // factorial = factorial*number;  factorial *=

      number;

      --number;

  }
  printf("Factorial= %lld", factorial);  return 0;
```

2.3 do...while loop

- **The do..while loop** is similar to the **while loop** with one important difference.
  - **The body of do...while loop** is executed once, before checking the test expression.
  - The do...while loop is executed at least once.

do...while loop Syntax

The syntax of a do while loop is:

```
do
{
   // codes
}
while (testExpression);
```

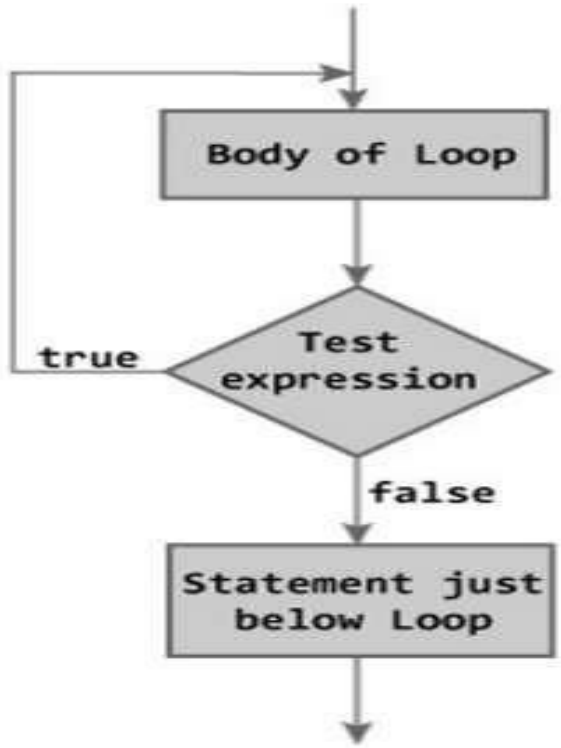


Figure: Flowchart of do...while Loop

Example: do...while loop

```
// Program to add numbers until user enters zero
#include <stdio.h>

int main() {
    double number, sum = 0;
    // loop body is executed at least once
    do{
        printf("Enter a number: ");  scanf("%lf",
        &number);  sum += number;
    }while(number != 0.0);   printf("Sum = %.2lf",sum);
    return 0;
    }
```

2.4 Nested loops

- C programming allows to use one loop inside another loop

- **Syntax for loop**

```
for ( init; condition; increment ) {

  for ( init; condition; increment ) {
    statement(s);
  }

  statement(s);
}
```

2.4 Nestedloops (Con..)

- **Syntax while loop**

```
while(condition) {

  while(condition) {

  statement(s);
  }

  statement(s);
}
```

3. Break And Continue Statement

- What is BREAK meant?
- What is CONTINUE meant?

Syntax of if...else if....else statement.

- **The break statement** terminates the loop immediately when it is encountered.

- The break statement is used with **decision making statement such as if...else**.

- **Syntax of break statement**

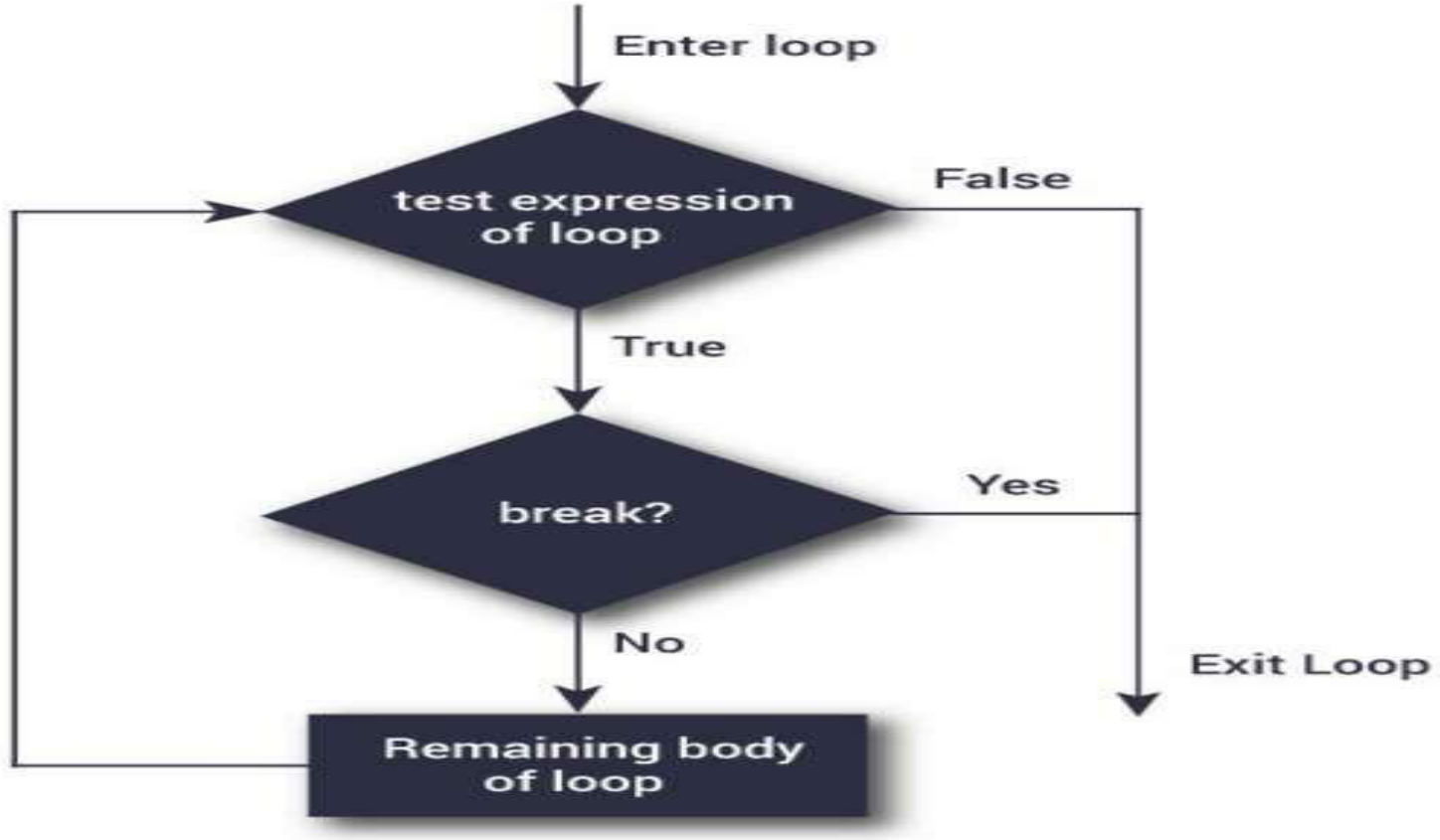
break;

How break statement works?

```
while (test Expression)
{
        // codes
        if (condition for break)
        {
                break;
        }
        // codes
}
```

```
for (init, condition, update)
{
        // codes
        if (condition for break)
        {
                break;
        }
        // codes
}
```

Flowchart Of Break Statement



Example: break statement

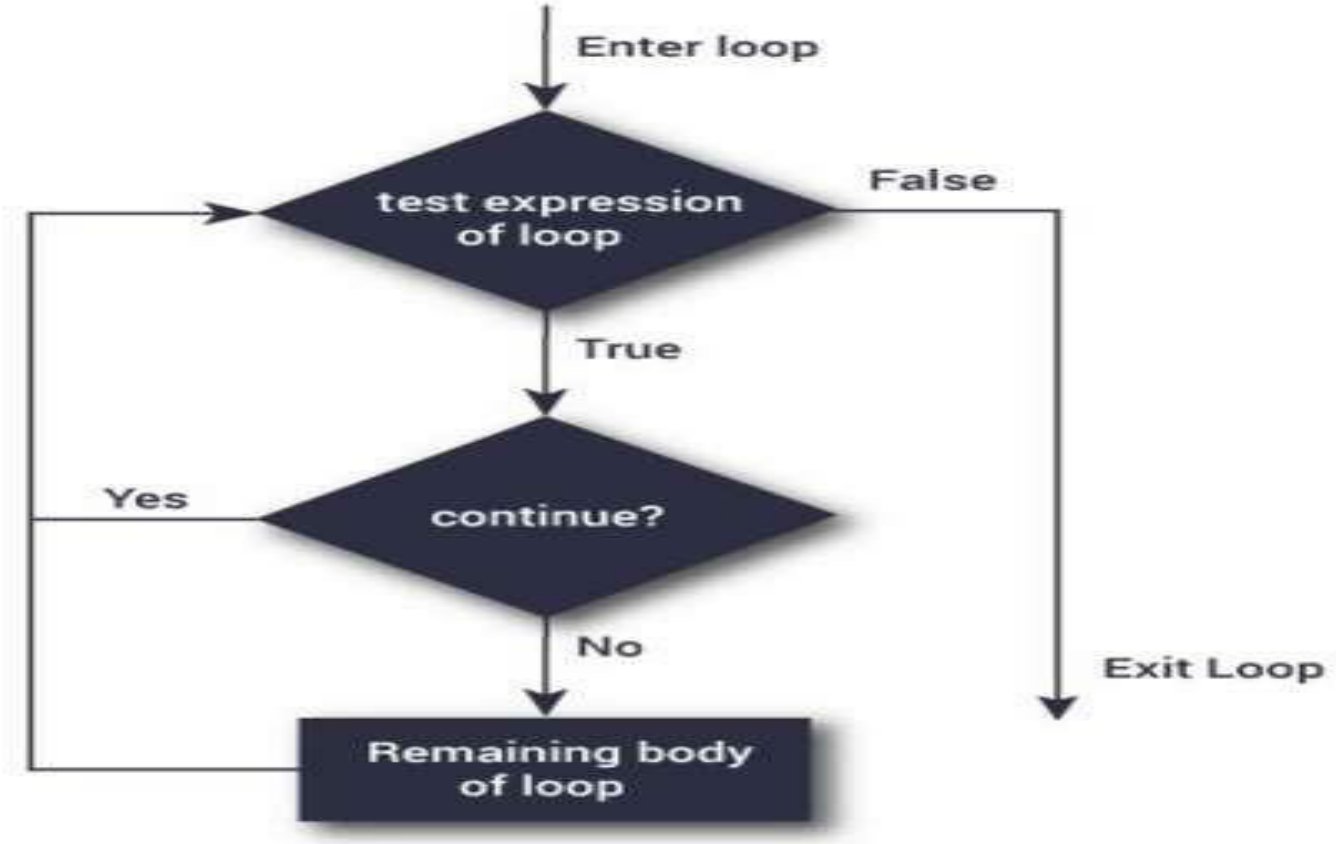
```
// Program to calculate the sum  of
maximum of 10 numbers
// Calculates sum until user enters
positive number
 # include <stdio.h>

int main() {

    int i;
    double number, sum = 0.0;

    for(i=1; i <= 10; ++i) {

        printf("Enter a n%d: ",i);

        scanf("%lf",&number);
```

```
// If user enters negative number,
loop is terminated
if(number < 0.0) {

        break;

    }
    // sum = sum + number;

    sum += number;

}
printf("Sum = %.2lf",sum);
return 0;
}
```

3.2 Continue Statement

- **The continue statement** skips some statements inside the loop.
- The continue statement is used with **decision making statement such as if... else**.
- **Syntax of continue Statement**
  - **continue;**

Flowchart of Continue Statement



How Continue Statement Works?

```
while (test Expression)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```

```
for (init, condition, update)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```

Example: continue statement

```
// Program to calculate sum of maximum of 10 numbers
// Negative numbers are skipped from calculation
# include <stdio.h>
int main(){
    int i;
    double number, sum = 0.0;
    for(i=1; i <= 10; ++i) {
        printf("Enter a n%d: ",i);
        scanf("%lf",&number);
```

```
// If user enters negative number, loop is terminated
        if(number < 0.0) {
            continue;
        }
    // sum = sum + number;
        sum += number;
    }
    printf("Sum = %.2lf",sum);
    return 0;
}
```

4. Switch Statement

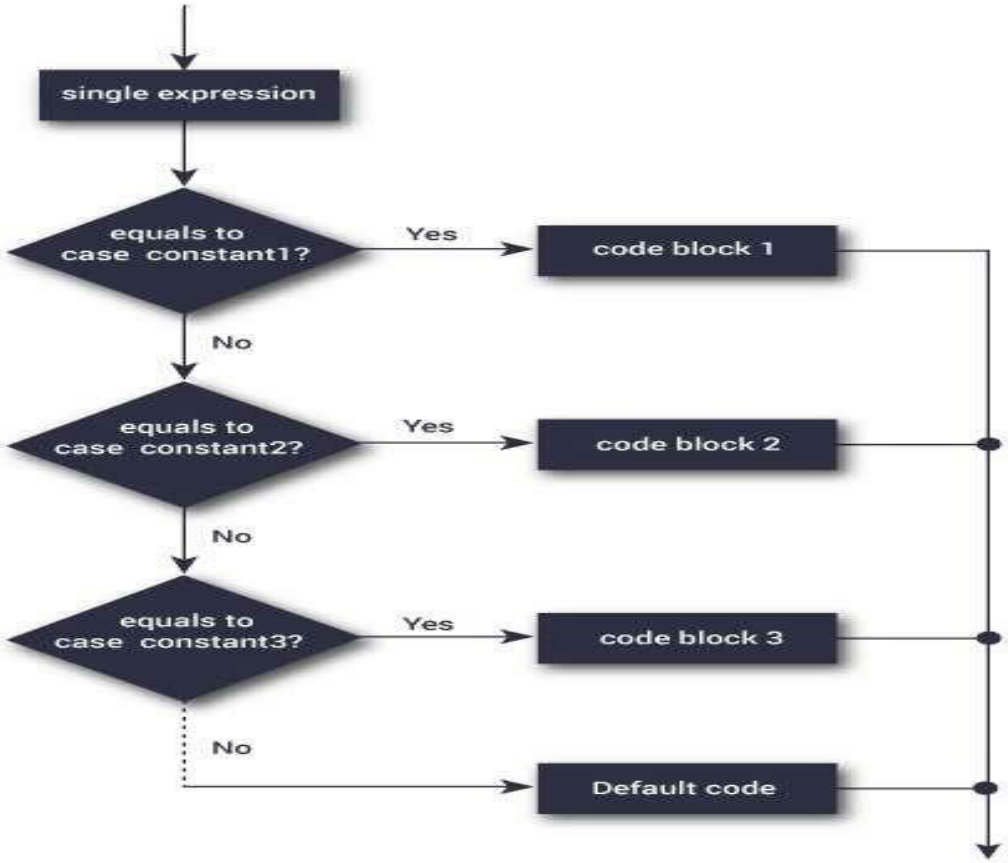
- The **if...else if…else statement** allows you to execute a block code among many alternatives. If you are checking on the value of a single variable in **if...else if…else statement**, it is better to use **switch statement**.
- **The switch statement** is often faster than nested if...else **(not always)**.

  Also, the syntax of switch statement is cleaner and easy to understand.

Syntax of switch...case

```
switch (n){
    case constant1:
        // code to be executed if n is equal to constant1;  break;

    case constant2:
        // code to be executed if n is equal to constant2;  break;

        ....
    default:

        // code to be executed if n doesn't match any constant

        }
```

Switch Statement Flowchart



Example: switch Statement

```
// Program to create a simple calculator

// Performs addition, subtraction, multiplication or division
  depending the input from user
  # include <stdio.h>

  int main() {

      char operator;
      double firstNumber,secondNumber;  printf("Enter an

      operator (+, -, *,): ");

      scanf("%c", &operator);  printf("Enter two operands: ");

      scanf("%lf %lf",&firstNumber, &secondNumber);
```

Syntax of if...else if....else statement.

```
switch(operator) {  case '+':
            printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber, firstNumber+secondNumber);  break;

        case '-':
            printf("%.1lf - %.1lf =  %.1lf",firstNumber, secondNumber, firstNumber-secondNumber);  break;

        case '*':
            printf("%.1lf * %.1lf = %.1lf",firstNumber, secondNumber, firstNumber*secondNumber);  break;

        case '/':
            printf("%.1lf / %.1lf = %.1lf",firstNumber, secondNumber, firstNumber/firstNumber);  break;

        // operator is doesn't match any case constant (+, -, *, /)

        default:

            printf("Error! operator is not correct");

    }
    return 0; }
```

5. goto Statement

- The goto statement is used to **alter** the normal sequence of a C program.

Syntax of goto Statement

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

What is Label?

- **The label** is an identifier. When **goto statement** is encountered, control of the program jumps to **label:** and starts executing the code.

```
goto label;
... .. ...
... .. ...

label:
... .. ...
... .. ...
```

Example: goto Statement

// Program to calculate the sum and average of maximum of 5 number
// If user enters negative number, the sum and average of previously entered positive number is displayed

```
# include <stdio.h>

int main(){

    const int maxInput = 5;  int i; double number, average, sum=0.0;

for(i=1; i<=maxInput; ++i){

        printf("%d. Enter a number: ", i);  scanf("%lf",&number);
```

Example: goto Statement

```
// If user enters negative number, flow of program moves  to label jump
        if(number < 0.0)  goto jump;


        sum += number; // sum = sum+number;
    }
    Jump:
    average=sum/(i-1);
    printf("Sum = %.2f\n", sum);  printf("Average = %.2f", average);  return 0; }
```


Thank
You