

CONTROL STATEMENTS IN C

Control Statements in C are used to perform actions like branching or looping.

A C program may require that a logical test be carried out at some particular point within the program. Depending on the outcome of the logical test, one of several possible actions will be carried out. This is known as branching. Eg: if-else, Nested if-else, switch statements.

A program may require that a group of instructions be executed repeatedly, until some logical condition has been satisfied. This is known as looping. Eg: for, while, do-while

if statement

The if statement evaluates the test expression inside the parenthesis.

The syntax of if statement is:

```
if (testExpression)
{
    statement 1;
    statement 2;
}
```

If the test expression is evaluated to true (nonzero), statements inside the body of "if" is executed. If the test expression is evaluated to false (0), statements inside the body of "if" is skipped from execution.

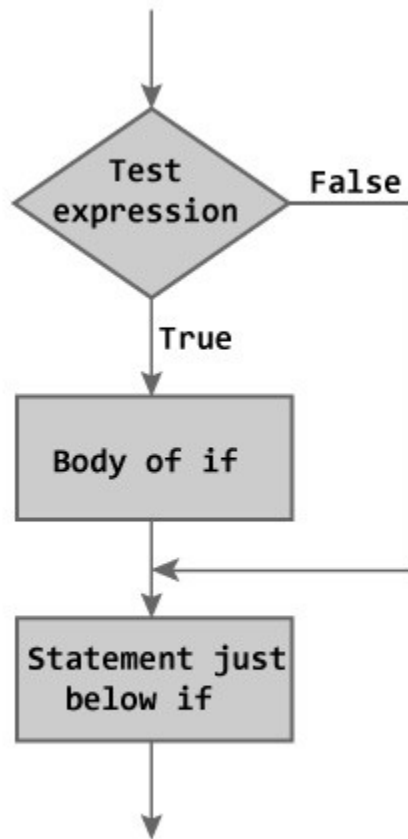


Figure: Flowchart of if Statement

Example

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

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

    if (number < 0)
    {
        printf("You entered %d\n", number);
    }

    return 0;
}
```

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).

The syntax of if-else statement is:

```
if (testExpression)
{
    // code inside the body of if
}
else
{
    // code inside the body of else
}
```

If test expression is true, codes inside the body of if statement is executed and, codes inside the body of else statement is skipped.

If test expression is false, codes inside the body of else statement is executed and, codes inside the body of if statement is skipped.

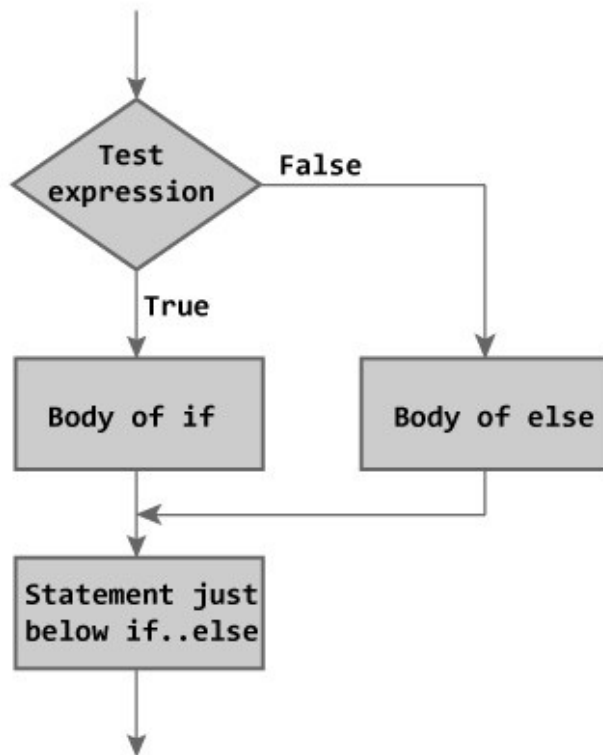


Figure: Flowchart of if...else Statement

Example

```
#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;
}
```

NESTED IF-ELSE STATEMENT

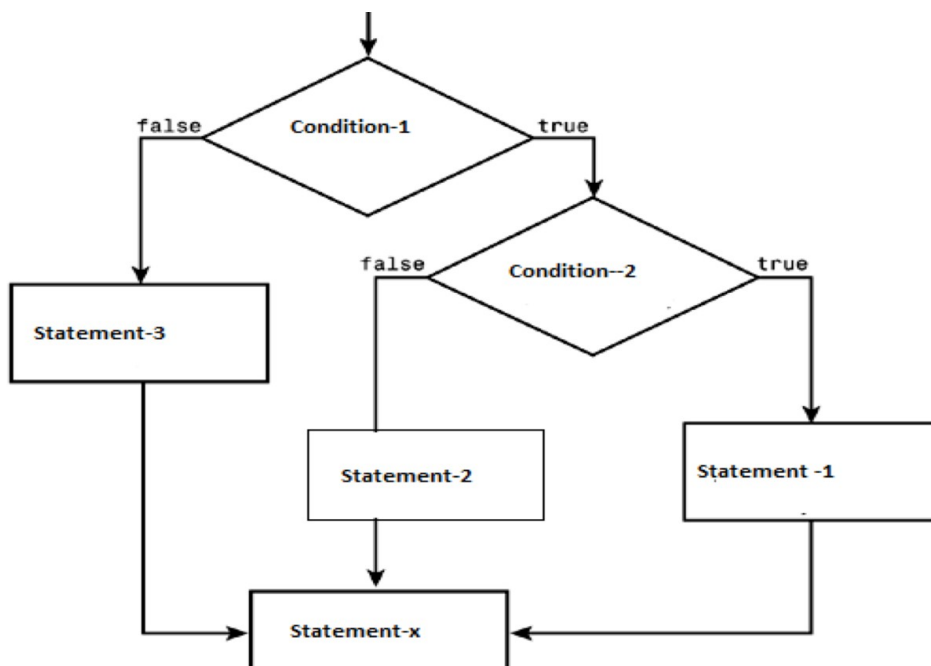
When a series of decisions are involved, more than one if...else statement is used in nested form.

Syntax of Nested if-else statement

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

statement-x;

if test-condition-1 is false, statement 3 will be executed, otherwise, test-condition 2 is evaluated. If test-condition 2 is true, statement-1 is evaluated, otherwise statement -2 will be evaluated and then control is transferred to statement -x.



PROGRAM FOR NESTED IF... ELSE STATEMENT:

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a,b,c;
    printf("enter three values\n");
    scanf("%d%d%d",&a,&b,&c);

    if(a>b)
    {
        if(a>c)
            printf("%d\n",a);
        else
            printf("%d\n",c);
    }
    else
    {
        if(c>b)
            printf("%d\n",c);
        else
            printf("%d\n",b);
    }
    return 0;
}
```

Output

```
enter three values
5 8 3
8
```

The ELSE IF Ladder

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

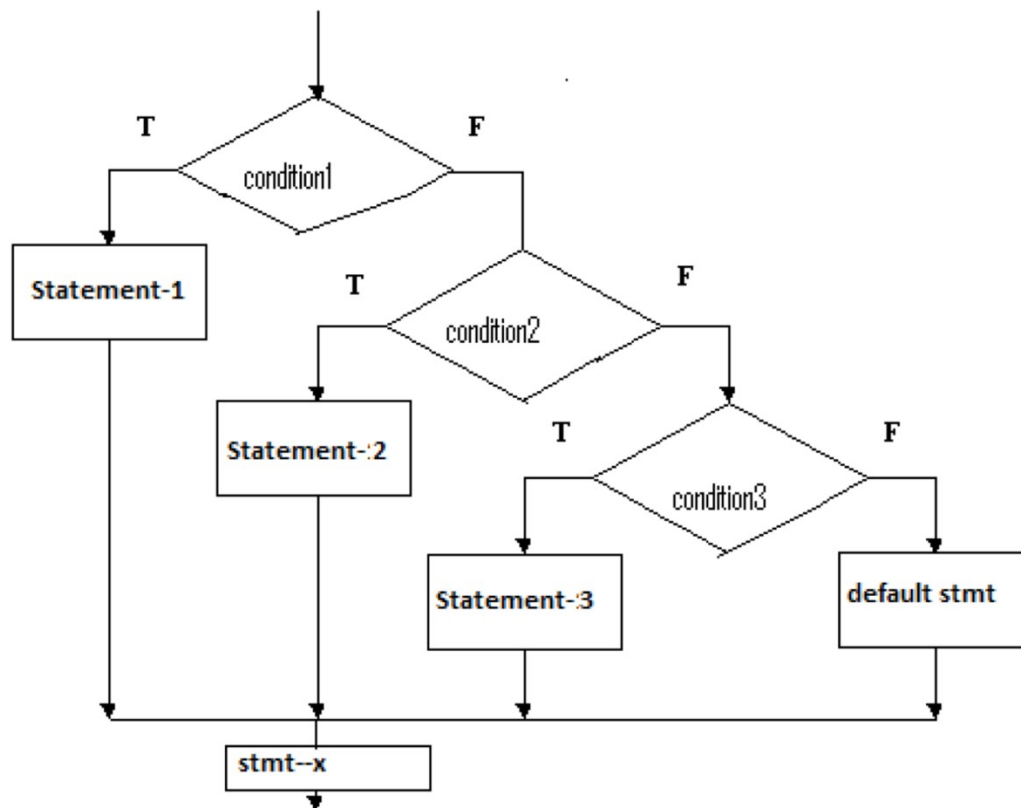
Syntax of else-if ladder statement

```
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
{
    default stmt;
    // statements to be executed if all test expressions are false
}

```



PROGRAM FOR THE ELSE IF LADDER STATEMENT:

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int m1,m2,m3, tot;
    float avg;
    printf("enter three subject marks");
    scanf("%d%d%d", &m1,&m2,&m3);
    tot=m1+m2+m3;
    avg=tot/3;
}

```

```
if (avg>=75)
{
    printf("distinction");
}
else if(avg>=60 && avg<75)
{
    printf("first class");
}
else if(avg>=50 && avg<60)
{
    printf("second class");
}
else if (avg<50)
{
    printf("fail");
}
return 0;
}
```

Output

enter three subject marks 83 89 95
distinction

LOOPS

Loops are used in programming to repeat a specific block of code.

for loop

The syntax of for loop is:

```
for (initializationStatement; testExpression; updateStatement)
{
    statement 1;
    statement 2;
}
```

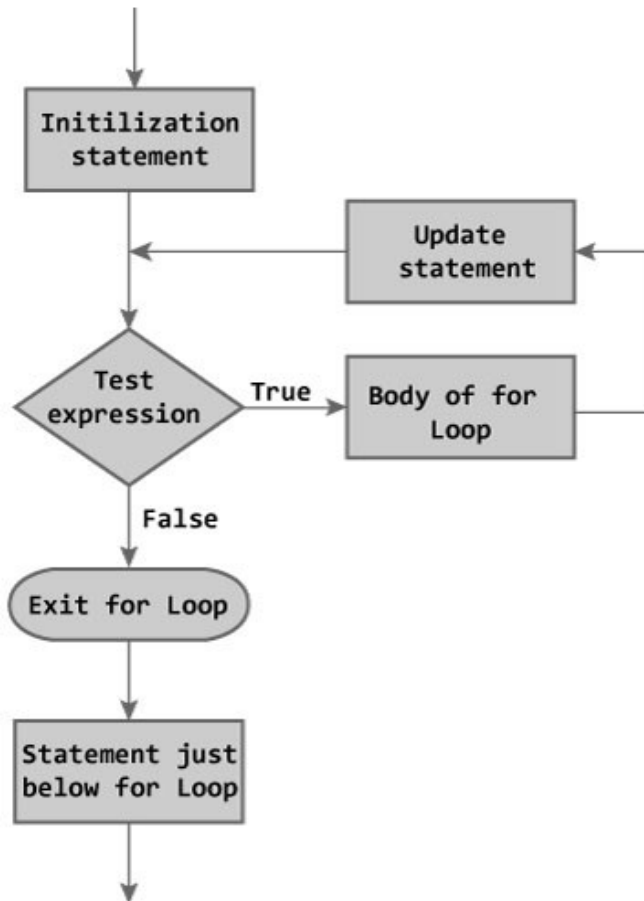



Figure: Flowchart of for Loop

Example to find the sum of integers from 1 to a value entered by user

OR Sum of first n 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 = sum + count;
```

```
    }
```

```
    printf("Sum = %d", sum);
```

```
    return 0;
```

```
}
```

count=1

count=5

count=2

count=6

count=4

sum=0

sum=sum+count=0+1=1

sum=sum+count=1+2=3

sum=sum+count=3+3=6

sum=sum+count=6+4=10

sum=sum+count=10+5=15

while Loop:

The while loop evaluates the test expression.

If the test expression is true (nonzero), code inside the body of while loop are executed. The test expression is evaluated again. The process goes on until the test expression is false. When the test expression is false, the while loop is terminated.

while loop is used when a number of times the loop is to be executed is not known in advance.

The syntax of while loop is

Initialization statement;

```
while (test expression)
{
    statement 1;
    statement 2;
    update statement;
}
```

```
count=1;
while (count <=num)
{
    sum+=count;
    count++;
}
```

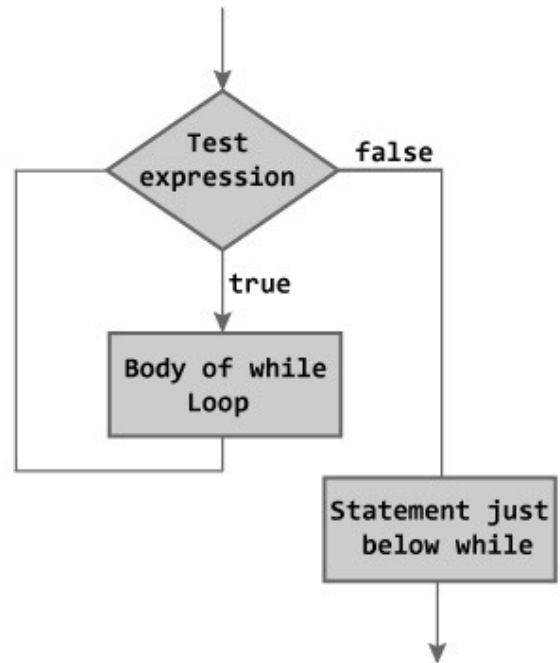


Figure: Flowchart of while Loop

Example:

```
#include <stdio.h>
int main()
{
    int n,i;
    long f;

    printf("enter an integer");

    scanf("%d", &n);
    f=1;
    i = 1;
    while (i <=n) // for (i=1,f=1; i<=n; i++)
    {
        f=f*i;
        i++;
    }
    printf("factorial is %ld", f);
    return 0;
}
```

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. Hence, the do...while loop is executed at least once.

Syntax for do while is

```
do
{
statement 1;
statement 2;
}
while (test expression);
```

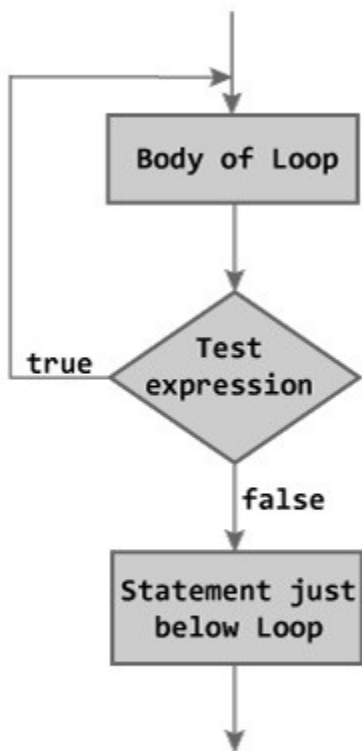


Figure: Flowchart of do...while Loop

Example to find the sum of all the numbers entered until the user enters zero.

```

#include <stdio.h>
int main()
{
double n, sum;
sum=0;
do {
printf ("Enter a value");
scanf("%lf",&n);
sum+=n;
} while(n!=0)

printf("sum=%lf",sum);
return 0;
}

```

```

sum=0
n=4.5
sum=sum+n ;
sum=4.5
n=5
sum=9.5
n=0
sum=9.5

```

switch...case Statement

The if..else..if ladder 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, it is better to use switch statement.

```

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
}

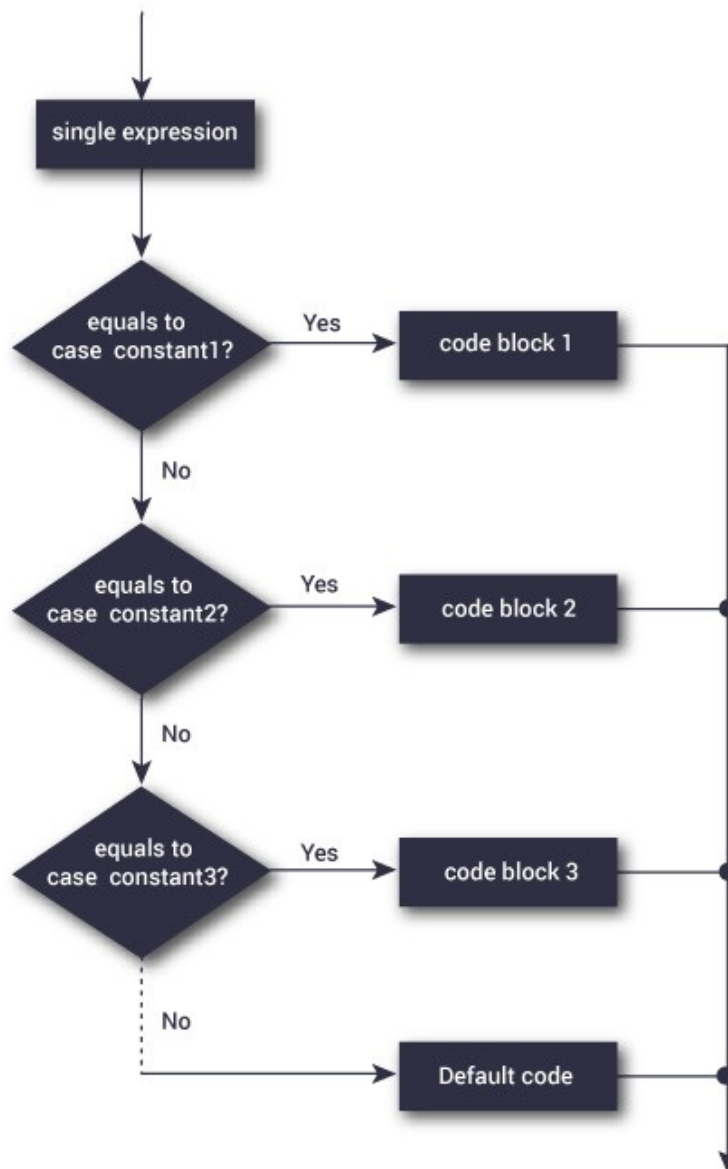
```

When a case constant is found that matches the switch expression, control of the program passes to the block of code associated with that case.

The compiler will execute the block of code associate with the case statement until the end of switch block, or until the break statement is encountered.

The break statement is used to prevent the code running into the next case.

switch Statement Flowchart



Example
include <stdio.h>
int main()

```

{
    char operator;
    double n1,n2;

    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);

    printf("Enter two operands: ");
    scanf("%lf %lf",&n1, &n2);

    switch (operator)
    {
        case '+':
            printf("%.1lf + %.1lf = %.1lf",n1, n2, n1+n2);
            break;

        case '-':
            printf("%.1lf - %.1lf = %.1lf",n1, n2, n1-n2);
            break;

        case '*':
            printf("%.1lf * %.1lf = %.1lf",n1, n2, n1*n2);
            break;

        case '/':
            printf("%.1lf / %.1lf = %.1lf",n1, n2, n1/n2);
            break;

        default:
            printf("Error! operator is not correct");
    }
    return 0;
}

```

operator

-

n1

4.3567

n2

2.6859

BREAK AND CONTINUE

It is sometimes desirable to skip some statements inside the loop or terminate the loop immediately without checking the test expression.

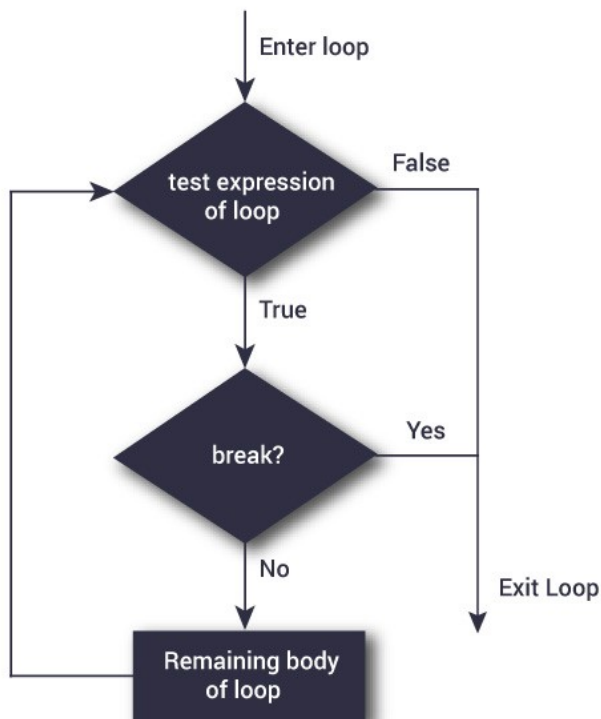
In such cases, break and continue statements are used.

break Statement

The break statement terminates the loop (for, while and do...while loop) immediately when it is encountered. The break statement is used with decision making statement such as if...else.

Syntax

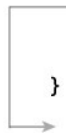
break;




```

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


```



```

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

```



Example program to enter 10 decimal numbers and break if user enters a zero

```

#include <stdio.h>
int main()
{
    int i;
    double number, sum;

    sum=0.0;
    for(i=1; i <= 10; ++i)
    {
        printf("Enter a number: ");
        scanf("%lf",&number);

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

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

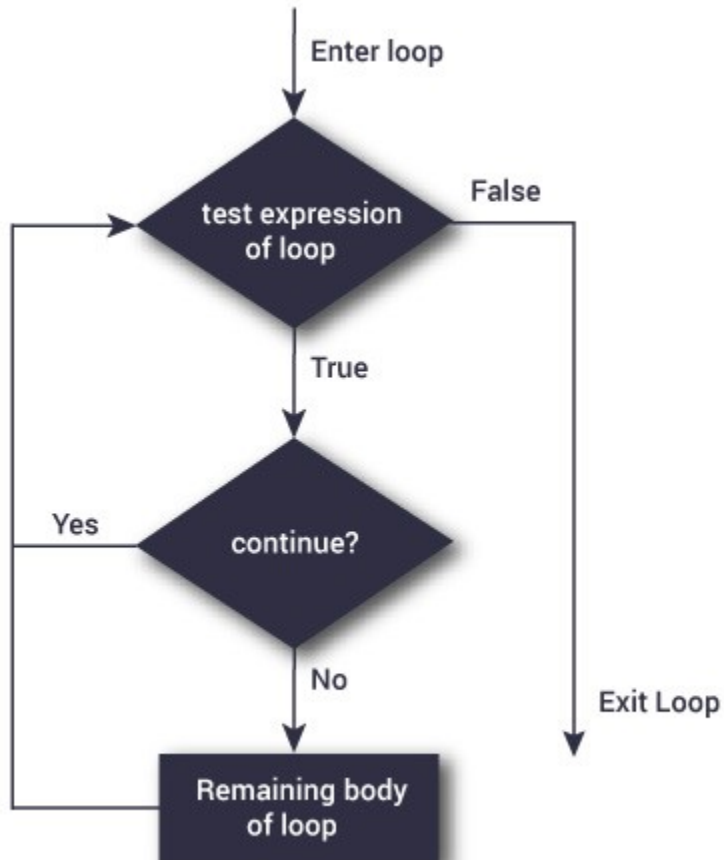
```

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

continue;



```

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

```

```

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

```

Example

```

#include <stdio.h>
int main()
{
    int i;
    double number, sum = 0.0;

    for(i=1; i <= 10; ++i)
    {
        printf("Enter a number: ");
        scanf("%lf",&number);

        // If user enters negative number, negative number is skipped from calculation
        if (number < 0.0)
        {
            continue;
        }

        sum += number; // sum = sum + number;
    }

    printf("Sum = %.2lf",sum);

    return 0;
}

```

Read a Natural Number and check whether the number is prime or not

```
#include <stdio.h>
int main()
{
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 2; i < n; i++)
    {
        if (n % i == 0)
        {
            flag = 1;
            break;
        }
    }

    if (n == 1)
    {
        printf("1 is neither prime nor composite.");
    }
    else
    {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }

    return 0;
}
```

OUTPUT

```
Enter a natural number: 2
2 is a prime number.

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter a natural number: 4
4 is not a prime number.

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter a positive integer: 1
1 is neither prime nor composite.
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Read a Natural Number and check whether the number is Armstrong or not

```
1  #include<stdio.h>
2  int main()
3  {
4      int n,r,sum=0,temp;
5      printf("Enter the number=");
6      scanf("%d",&n);
7      temp=n;
8      while(n>0)
9      {
10         r=n%10;
11         sum=sum+(r*r*r);
12         n=n/10;
13     }
14     if(temp==sum)
15         printf("armstrong  number ");
16     else
17         printf("not armstrong number");
18     return 0;
19 }
20
```

OUTPUT

```
Enter the number=153
armstrong  number

...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter the number=121
not armstrong number

...Program finished with exit code 0
Press ENTER to exit console.
```

Write a program to enter a number and print its reverse

```
1  #include<stdio.h>
2  int main()
3  {
4      int n, reverse=0, rem;
5      printf("Enter a number: ");
6      scanf("%d", &n);
7      while(n!=0)
8      {
9          rem=n%10;
10         reverse=reverse*10+rem;
11         n/=10;
12     }
13     printf("Reversed Number: %d",reverse);
14     return 0;
15 }
16
```

Enter a number: 456
Reversed Number: 654

...Program finished with exit code 0
Press ENTER to exit console.

8) Write a C program to find HCF (GCD) of two numbers.

```

1  #include <stdio.h>
2  int main()
3  {
4      int n1, n2, i, gcd;
5
6      printf("Enter two integers: ");
7      scanf("%d %d", &n1, &n2);
8
9      for(i=1; i <= n1 && i <= n2; ++i)
10     {
11         // Checks if i is factor of both integers
12         if(n1%i==0 && n2%i==0)
13             gcd = i;
14     }
15
16     printf("G.C.D of %d and %d is %d", n1, n2, gcd);
17
18     return 0;
19 }

```

Enter two integers: 2 3
G.C.D of 2 and 3 is 1

...Program finished with exit code 0
Press ENTER to exit console.

14) Write a C program to print Fibonacci series up to n terms.

Exercises

1) Read 3 integer values and find the largest among them using conditional operator.

- 2) Write a C program to find the sum of digits of an integer, entered through the keyboard
- 3) Write a C program to find the sum of all even numbers between two limits.
- 4) Write a C program to check whether a number is palindrome or not.
- 5) Write a C program to check if input character is a vowel using Switch Case.
- 6) Write a program to reverse the case of input character.

7) Addition and subtraction program using switch

- 1.Addition
- 2.Subtraction

Enter your choice : 1

Enter your numbers : 1 4

After Addition : 5

- 8) Program to find the list of odd and even digits in a number
- 9) Program to find the sum of odd digits in a given integer
- 10) Write a C program to check the number nearest to 500 among two given integers and the highest difference.
- 11) Write a program to accept an integer and print the sum of the last three digits as output
- 12) Write a program that prints all the numbers from 0 to 6 except 3 and 6 using continue statement

- 1) Program to print 2 numbers other than the largest number in 3 numbers.
- 2) Write a C program to find LCM of two numbers.
- 3) Write a C program to swap first and last digits of a number.
- 4) Write a program to read a character value and display its ASCII value
- 5) Print the multiples of 7 which are not multiples of 14 upto 5 numbers
21, 35, 49, 63, 70
- 6) Write a program to accept an integer in the range 0-5 and print the output in words.
- 7) Write a program to accept an integer n and calculate $n + nn + nnn$