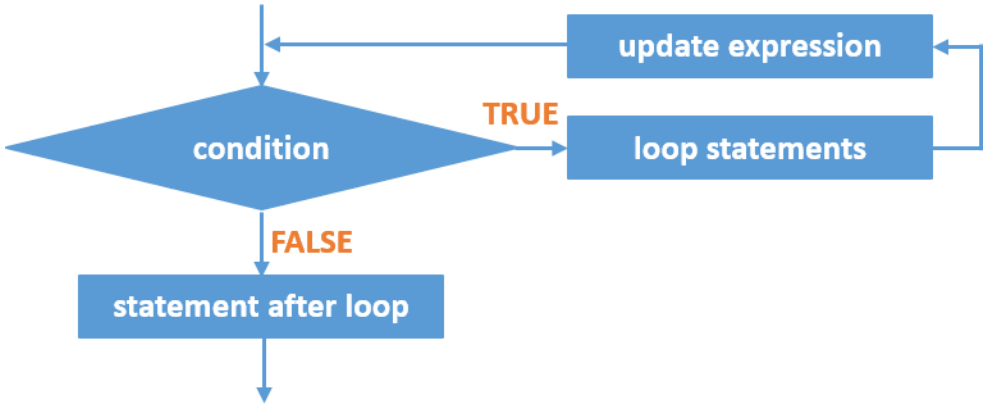
7. “while” Loop

A **while** loop is usually used to repeatedly execute a statement/block of statements as long as a given condition is TRUE (nonzero).

Syntax

```
while (condition)
{
   statements;
   ...
   update expression;
}
```



while Statement Flowchart

How it works?

1. The **condition** is evaluated. The **condition** may be a direct integer value, a variable, or an expression.
   - Any nonzero value is considered TRUE.
   - If the **condition** contains a variable, the variable must be initiated before it is used.
   - If the condition is TRUE, the statements within the loop will be executed including the update expression.
   - If the condition is FALSE, the loop statements will not be executed. Program control jumps to the next statement after **while** loop.
2. The condition is evaluated again. Step 2 is repeated until the condition becomes false or loop is terminated using break statement.

Example:

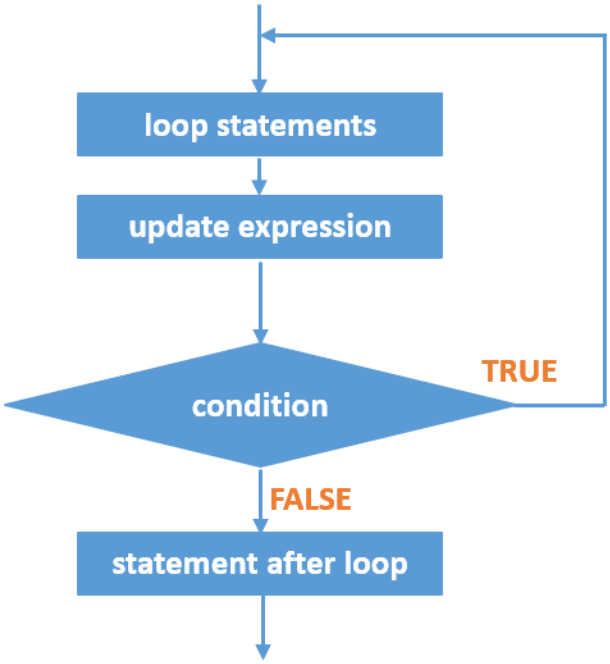
```
/* program finds sum of all integer numbers from 1 to n,
where n is entered by the user */
#include<stdio.h>

int main(){
    int n, counter=1, sum=0;
    printf("Enter integer number: ");
    scanf("%d", &n);
    while(counter <= n)
    {
        sum+= counter;
        counter++;
    }

    printf("Sum of integer numbers from 1 to %d = %d", n, sum);
    return(0);
}
```

"do-while" Loop

- A **do-while** loop is usually used to repeatedly execute a statement/block of statements as long as a given condition is TRUE (nonzero).
- The **condition** is tested after execution of the **do-while** loop's body A statement/block of statements within the **do-while** loop will be executed at least one time.

Syntax

```
do
{
   statements;
   ...
   update expression;
} while (condition);
```



do-while Statement
Flowchart

How it works?

3. Statements, including the update expression, within **do** block are executed.
4. The **condition** is evaluated. The **condition** may be a direct integer value, a variable, or an expression.
    - Any nonzero value is considered TRUE.
    - If the **condition** contains a variable, the variable must be initiated before it is used.
    - If the condition is TRUE, the statements within the loop will be executed again, including the update expression.
    - If the condition is FALSE, the program control jumps to the next statement after **do-while**
      loop.

Example:

```
/* program finds sum of all integer numbers from 1 to n,
where n is entered by the user */
#include<stdio.h>

int main(){
    int n, counter=1, sum=0;
    printf("Enter integer number: ");
    scanf("%d", &n);
    do
    {
        sum+= counter;
        counter++;
    } while(counter <= n);

    printf("Sum of integer numbers from 1 to %d = %d", n, sum);
    return(0);
}
```

References

- Tan, H.H., and T.B. D'Orazio. *C Programming for Engineering & Computer Science*. USA: WCB McGRaw-Hill. 1999. Print.
- Tutorialspoint.com. "While loop in C." *Www.tutorialspoint.com*. N.p., n.d. Web. 13 Mar. 2017. <https://www.tutorialspoint.com/cprogramming/c_while_loop.htm>.
- Tutorialspoint.com. "Do...while loop in C." *Www.tutorialspoint.com*. N.p., n.d. Web. 13 Mar. 2017. <https://www.tutorialspoint.com/cprogramming/c_do_while_loop.htm>.