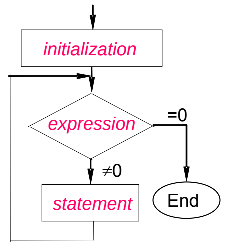
Loops

Loops (iteration statements) allow a set of instructions to be repeatedly executed until a certain condition is reached.

The condition may be predefined (for loop) or open ended (while and do-while loop).

while Loop

```
initialization
while (expression)
  statement
```



After *initialization* the *expression* is evaluated; if it is true (≠0), *statement* is executed and *expression* is re-evaluated. This cycle continues until *expression* becomes 0, at which point execution resumes after *statement*.

Example: The program copies its input to its output one character at a time.

```
read a character
while character is not end-of-file indicator
        output the character just read
        read a character
```

```
/* Copy input to output #1 */
#include <stdio.h>
int main()
{
   int ch;
   printf ("Enter sequence  of characters. For end press Ctrl/Z ");
   ch = getchar ();
   while (ch != EOF)
   {
      putchar (ch);
      ch = getchar ();
   }
   return 0;
}
```

```
/* Copy input to output #2 */
#include <stdio.h>
int main()
{
   int ch;                                          /* character */
   printf ("Enter sequence  of characters. For end press Ctrl/Z ");
   while ((ch = getchar ()) != EOF)
   {
      putchar (ch);
    }
   return 0;
}
```

Example: The program counts characters.

```
set number of characters to 0
read a character
while character is not end-of-file indicator
        increment number of characters
        read a character
print number of characters
```

```
/* Count characters in input */
#include <stdio.h>
int main()
{
   int nc;                                   /* number of characters */
   printf ("Enter sequence  of characters. For end press Ctrl/Z ");
   nc = 0;
   while (getchar () != EOF)
   {
      ++nc;
    }
    printf ("Number of characters = %d\n", nc);
   return 0;
}
```

Example: The program counts input lines. Each line in the sequence of lines terminates by a newline.

```
set number of lines to 0
read a character
while character is not end-of-file indicator
        if character is newline
                increment number of lines
        read a character
print number of lines
```

```
/* Count lines in input */
#include <stdio.h>
int main()
{
   int ch,                                   /* character          */
       nl;                                   /* number of lines    */
   printf ("Enter sequence  of lines. For end press Ctrl/Z ");
   nl = 0;
   while ((ch = getchar ()) != EOF)
   {
      if (ch == '\n')
         ++nl;
    }
    printf ("Number of lines = %d\n", nl);
   return 0;
}
```

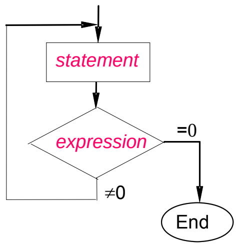
Example: The program counts lines, words, and characters. Word is any sequence of characters that does not contain a delimiter, i.e. white space (blank, tab or newline).

```
set state outside a word
set number of characters to 0
set number of words to 0
set number of lines to 0
read a character
while character is not end-of-file indicator
        increment number of characters
        if character is newline
                increment number of lines
        if character is a delimiter
                set state outside a word
        else if the state is outside a word
                set state inside a word
                increment number of words
        read a character
print number of lines
```

```
/* Count lines, words, and characters. */
#include <stdio.h>
#define IN 1                              /* inside a word             */
#define OUT 0                             /* outside a word            */
int main()
{
   int ch,                                /* character                 */
       nl,                                /* number of lines           */
       nw,                                /* number of words           */
       nc,                                /* number of characters      */
       state;                             /* whether the program is    */
                                          /* currently in a word or not */
   printf ("Enter sequence  of lines. For end press Ctrl/Z ");
   state = OUT;
   nl = nw = nc = 0;
   while ((ch = getchar ()) != EOF)
   {
     ++nc;
     if (ch == '\n')                              /* newline      */
        ++nl;
     if (ch == ' ' || ch == '\n' || ch == '\t')   /* delimiter    */
        state = OUT;
     else if (state == OUT)           /* first character of a word */
     {
        state = IN;
        ++nw;
     }
    }
    printf ("Lines = %d\nWords = %d\nCharacters = %d\n",
             nl, nw, nc);
   return 0;
 }
```

do-while Loop

```
do
   statement
while (expression);
```



The *statement* is executed, then *expression* is evaluated; if it is true (≠0), *statement* is evaluated again, and so on. When the *expression* becomes false (0), the loop terminates.

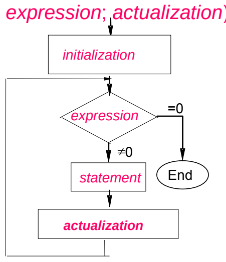
Example: The program tests whether the input integer number is in the given region [1,100].

```
do
        input number
while number is out of range
```

```
/* Test the validity of the input number in the region 1 – 100. */
#include <stdio.h>
int main()
{
   int number;
   do
   {   printf ("Enter number between 1 and 100: ");
       scanf ("%d", &number);
   }
   while (number<1 || number>100);
   return 0;
}
```

for Loop

```
for (initialization; expression; actualization)
   statement
```



After *initialization* the *expression* is evaluated; if it is true (≠0), *statement* is executed, *actualization* is evaluated and *expression* is re-evaluated. This cycle continues until *expression* becomes 0, at which point execution resumes after *statement*.

Any of the three parts can be omitted, although the semicolons must remain.

Infinite loop

```
for (; ;)
{
   . . .
}
```

or

```
while (1)
{
   . . .
}
```

Example: The program calculates the sum of a sequence of integers with a given number.

```
enter the given number
set the sum to 0
set the counter to 1
while the given number is not reached
   input an integer
   add integer to the sum
   increment the counter
print the sum
```

```c
/* Calculate sum of a given number of integers */
#include <stdio.h>
int main()
{
   int n,                              /* number of integers         */
        number,                        /* integer number             */
        count,                         /* counter                    */
        sum;                           /* sum of integers            */
   printf ("Enter number of integers: ");  scanf("%d", &n);
   sum = 0;
   for (count = 1; count <= n; count++)
   {  printf ("Enter integer: ");
      scanf("%d", &number);
      sum += number;
   }
   printf("Sum of integers is %d.\n", sum);
   return 0;
}
```