Computer science : c++

do-while loop in C++

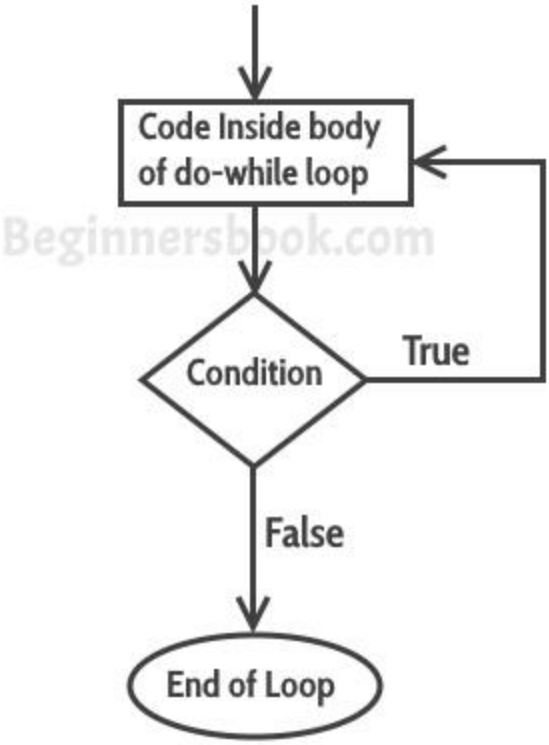
. do-while loop is similar to while loop, however there is a difference between them: In while loop, condition is evaluated first and then the statements inside loop body gets executed, on the other hand in do-while loop, statements inside do-while gets executed first and then the condition is evaluated.

Syntax

```
do
{
   statement(s);
} while(condition);
```

WORKING OF do-while loop

First, the statements inside loop execute and then the condition gets evaluated, if the condition returns true then the control jumps to the “do” for further repeated execution of it, this happens repeatedly until the condition returns false. Once condition returns false control jumps to the next



PROGRAMMING IN do-while loop

```
#include <iostream>
using namespace std;
int main(){
   int num=1;
   do{
      cout<<"Value of num: "<<num<<endl;
      num++;
   }while(num<=6);
   return 0;
}
```

Output:

```
Value of num: 1
Value of num: 2
Value of num: 3
Value of num: 4
Value of num: 5
Value of num: 6
```

ARRAY ELEMENTS USING DO-WHILE LOOP

Here we have an integer array which has four elements. We are displaying the elements of it using do-while loop.

```
#include <iostream>
using namespace std;
int main(){
   int arr[]={21,99,15,109};
   /* Array index starts with 0, which
    * means the first element of array
    * is at index 0, arr[0]
    */
   int i=0;
   do{
      cout<<arr[i]<<endl;
      i++;
   }while(i<4);
   return 0;
}
```

Output:

```
21
99
15
109
```

Continue Statement in C++ with example

Continue statement is used inside loops. Whenever a continue statement is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration, skipping the execution of statements inside loop's body for the current iteration.

Syntax

```
continue;
```

CONTINUE STATEMENT

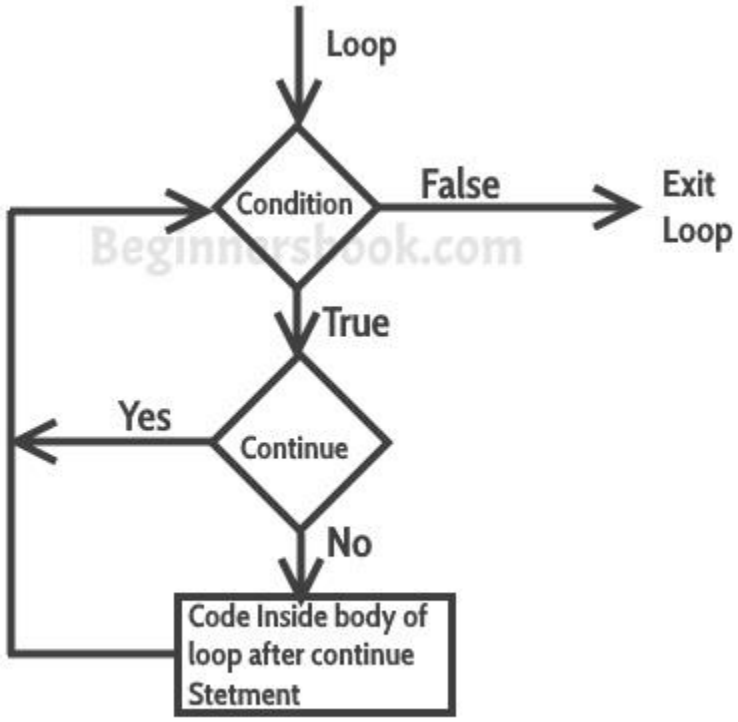
As you can see that the output is missing the value 3, however the for loop iterate though the num value 0 to 6. This is because we have set a condition inside loop in such a way, that the continue statement is encountered when the num value is equal to 3. So for this iteration the loop skipped the cout statement and started the next iteration of loop.

```
#include <iostream>
using namespace std;
int main(){
   for (int num=0; num<=6; num++) {
      /* This means that when the value of
       * num is equal to 3 this continue statement
       * would be encountered, which would make the
       * control to jump to the beginning of loop for
       * next iteration, skipping the current iteration
       */

      if (num==3) {
          continue;
      }
      cout<<num<<" ";
   }
   return 0;
}
```

Output: 0 1 2 3 4 5 6

Flow CHART



Use of continue in While loop

```
#include <iostream>
using namespace std;
void main(){
   int j=6;
   while (j >=0) {
      if (j==4) {
         j--;
         continue;
      }
       cout<<"Value of j: "<<j<<endl;
       j--;
   }
}
```

Output:

```
Value of j: 6
Value of j: 5
Value of j: 3
Value of j: 2
Value of j: 1
Value of j: 0
```

continue in do-While loop

```
#include <iostream>
using namespace std;
int main(){
   int j=4;
   do {
      if (j==7) {
         j++;
         continue;
      }
      cout<<"j is: "<<j<<endl;
      j++;
   }while(j<10);
   return 0;
}
```

Output:

```
j is: 4
j is: 5
j is: 6
j is: 8
j is: 9
```