

#7 Iterative Control Instruction - Loop

By Saurabh Shukla | 2017©mysirg.com

Iterative Control Instructions

Iterative control instruction is also known as repetitive control instruction or loop. It is used to execute a piece of instructions again and again. This can reduce the efforts of writing similar code multiple times. Also it can be used to manage logical sequence of operations.

In c language, loops can be implemented with the following three ways:

- while
- do while
- for

while loop

Following is the syntax of while loop.

```
...  
while(some condition){  
  
    statement1;  
    statement2;  
    ...  
}  
...
```

- while is a keyword.
- It looks like if statement but different on one thing, control comes back to while after reaching end of while block.
- while block is set of statements bounded by pair of curly braces and present immediately after the while(some condition).
- Parenthesis followed by while keyword can never be empty
- There is no semicolon after while (condition), because while statement is not terminating there.
- Whatever you have written in parenthesis is evaluated as true or false
- If the condition in the parenthesis is false, control simply skipped while block.
- If the condition in the parenthesis is true, control enters in while block and execute statements present in the while block. When control reaches at the end of while block it again goes back to while keyword and recheck the condition written in the parenthesis.

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

- If the condition in the parenthesis is again true, control again enters in while block. This keeps on going until condition becomes false.
- You can say while loop as entry control loop, because control can enter in the while block only after checking the condition.
- You want to control the iteration, for which you have to make condition in such a way that it changes on iterations.
- Assume, you want to print mysirg five times on the screen. You can do it either by writing printf five times or by using loop. Loop is the smarter approach. It reduces your efforts of writing printf multiple times. You can write printf only once and iterate it for five times.

```
1  int main()  
2  {  
3      int i=1;  
4      while(i<=5){  
5          printf("mysirg\n");  
6          i++;  
7      }  
8  }  
9  
10
```

- In the above program, we have used variable i, which is initialized by 1.
- Control then comes to while statement. Condition $i \leq 5$ should be read as '1 less than or equal to 5', which is true. Control enters into the while block, printing mysirg for the first time. On 6th line $i++$; the value of i is increased to 2. Control reaches to the end of while block. From there it again moves to while's condition $i \leq 5$, but due to the change in value of i, we now read it as '2 less than or equal to 5'.
- Keep moving control in this way, at one point of time, value of i becomes 5 and 'mysirg' is printed for the 5th time, then i is incremented to 6. Control reaches at the end of while block and from there it goes again to the condition $i \leq 5$; read it as '6 is less than or equal to 5', which is false. Control simply skipped the while's body.
- Notice that the condition is checked for 6 times (5 times true, 1 time false).
- $i++$; is an important step, otherwise looping never ends.
- Initialization ($i=1$), condition ($i \leq 5$) and flow ($i++$) are logical constructs to control the number of iteration.

do while loop

Following is the syntax of do while loop

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

```
...
do{

    statement1;
    statement2;
    ...
}while(some condition);
...
```

- do and while are keywords
- do while is exit control loop
- Since while(condition) is written at the end of the statement, it is followed by a semicolon as a mark of termination of the statement.
- Control when reaches 'do' keyword, it simply enters into the do-while block. Where it executes statements and then reaches to the end of block. Just after the block while(condition) is encountered. Where condition is evaluated either as true or false. If it is true, control moves back to the keyword 'do' and repetition begins. If condition is false, it simply move forward to the next statement of the program.
- In do while loop, even if the condition is false for the first time, then also control is able to visit do while block once. This is why it is called exit control loop.
- Do while loop guarantees running of code at least once
- Previous program of printing 'mysirg' 5 times on the screen can be rewritten using do while loop as:

```
1  int main()
2  {
3      int i=1;
4      do{
5          printf("mysirg\n");
6          i++;
7      }while(i<=5);
8
9  }
10
```

- Dry run above program to figure out the difference between working of while and do while loop.
- One difference you can notice that in do while loop condition is checked for 5 times (in the above program). First four times it is true then 5th time it is false. First time condition $i \leq 5$ should be read as '2 less than or equal to 5'. The value of i is 2 because it control already visited do while block before condition checking. On the other hand, in while loop condition is checked for 6 times.

for loop

Following is the syntax of for loop

```
...
for(initialization; condition; flow){

    statement1;
    statement2;
    ...
}
...
```

- for is a keyword
- The most appealing feature of for loop is its format which is capable to provide place three basic logical constructs to control looping, initialization, condition and flow at one place.
- In the for's parenthesis, exactly two semicolons are mandatory.
- Whatever written before the first semicolon is executed first, then control moves to condition part, which is written between the semicolons. Condition is evaluated as true or false. If it is false, control simply skipped for's block. If the condition is true control enters into the for's block. After executing statements residing in the for's block, control jumps up to the flow section, which is written after the second semicolon in the parenthesis. Control again moves to the condition part and process repeats.
- The for loop is also an entry control loop.
- Following is the same program of printing mysirg five times on the screen but with the for loop.

```
1  int main()
2  {
3      int i=1;
4      for(i=1;i<=5;i++){
5          printf("mysirg\n");
6      }
7  }
```

- Dry run the above code according to the rules just described.
- You can remove curly braces of for's body, as it contains only one statement.

Keyword break

- break is a keyword
- It can only be used either in the body of loop or in the body of switch

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

- The keyword break is used in the body of loop to terminate execution of loop before completion of its normal life.
- Sometimes it is desired to terminate loop before its normal end.
- For example we have to make a program to take input a number from user, which should be an even number. We give at the most three chances to the user to input correct value. It is like a game where user has three chances. If user entered an even number the game is in his pocket (win the game) otherwise he loses the game. Suppose user inputs an even number in the first chance, then no more chances are needed as he already wins the game. So we have to use break keyword to terminate the iterations.

```
1  int main()  
2  {  
3      int x,i=1;  
4      while(i<=3)  
5      {  
6          printf("Enter an even number");  
7          scanf("%d",&x);  
8          if(x%2==0)  
9          {  
10             printf("You Win");  
11             break;  
12         }  
13         i++;  
14     }  
15     if(i==4)  
16         printf("You lost");  
17     return(0);  
18 }  
19
```

- Dry run above program. As soon as break keyword encounters, control immediately comes out of the loop body.
- We put break keyword inside if statement because we want to make it happen only when user input is divisible by 2.
- You cannot use break outside the loop body.
- You will see more about keyword break in switch case control instruction in the next chapter.

Keyword Continue

- The keyword continue is used only in the body of loop.
- It is used to move control at termination condition in the case of while and do-while loop.
- continue is used in for loop to transfer control at flow part.

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

```
1  int main()
2  {
3      int x;
4      while(1)
5      {
6          printf("Enter an even number");
7          scanf("%d",&x);
8          if(x%2==1)
9              continue;
10         else
11         {
12             printf("This is the correct value");
13             break;
14         }
15     }
16     return(0);
17 }
18
```

- In the above program, condition of while loop is always evaluated as TRUE. Any non-zero number is considered as TRUE and zero is considered as FALSE.
- The loop only ends at the execution of break.
- If the user enters an odd number condition $x\%2==1$ becomes TRUE, then continue works.
- Continue transfers the control at while(1).
- As long as user keeps on entering odd numbers, continue works, but once the user enters an even number, break will terminate loop.

Remember

- You can have loop inside a loop, it is called nested loop. For example

```
1  main()
2  {
3      int i,j;
4
5      for(i=1;i<=4;i++){
6
7          for(j=1;j<=3;j++){
8
9              printf("i=%d j=%d ",i,j);
10
11          }
12          printf("\n");
13      }
14  }
15
```

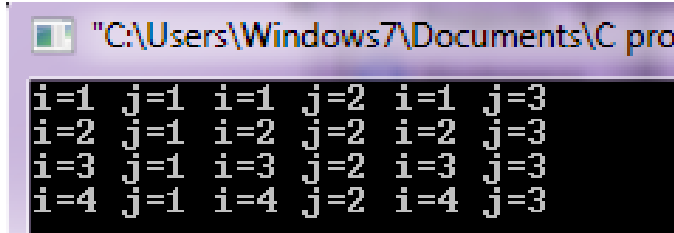
- In the above example `printf("i=%d j=%d ",i,j);` will execute 12 times. Do dry run to understand its execution. When i is 1, inner loop will run for 3 times and prints
`i=1 j=1 i=1 j=2 i=1 j=3`

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

then printf("\n") moves the cursor to the next line. Control goes back to the outer for loop and increment value of i. This keeps on repeating for 4 times. Output of the program:



```
"C:\Users\Windows7\Documents\C pro"
i=1 j=1 i=1 j=2 i=1 j=3
i=2 j=1 i=2 j=2 i=2 j=3
i=3 j=1 i=3 j=2 i=3 j=3
i=4 j=1 i=4 j=2 i=4 j=3
```

- You can leave parenthesis of for loop blank (two semicolons must be there), it is considered as infinite loop.
- You cannot leave parenthesis of while empty.

References

YouTube video links

- Lecture 7 Loops in C part-1
 - <https://www.youtube.com/watch?v=gXEoxT-v1Fc&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW>
- Lecture 7 Loops in C part-2
 - <https://www.youtube.com/watch?v=x-1qJpL5uUw&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW>
- Lecture 7 Loops in C part-3
 - https://www.youtube.com/watch?v=NKRQDN_QZ8w&feature=youtu.be&list=PL7ersPsTyYt2Q-SqZxTA1D-melSfqBRMW

Exercise

1. Write a program to print mysirg 10 times.
2. Write a program to print first 10 natural numbers.
3. Write a program to print first 10 natural numbers in reverse order.
4. Write a program to print first 100 natural numbers, 10 numbers in each line.
5. Write a program to print first N natural numbers.
6. Write a program to print first N natural numbers in reverse order
7. Write a program to print first 10 odd natural numbers.
8. Write a program to print first 10 odd natural numbers in reverse order.
9. Write a program to print first N odd natural numbers.
10. Write a program to print first N odd natural numbers in reverse order
11. Write a program to print first 10 even natural numbers.
12. Write a program to print first 10 even natural numbers in reverse order.
13. Write a program to print first N even natural numbers.
14. Write a program to print first N even natural numbers in reverse order.
15. Write a program to print table of 5.

#7 Iterative Control Instruction-Loop

C Notes Vol-1 by Saurabh Shukla

www.mysirg.com

16. Write a program to print table of N.
17. Write a program to calculate sum of first N natural numbers.
18. Write a program to calculate product of first N natural numbers.
19. Write a program to calculate factorial of a number.
20. Write a program to calculate sum of first N odd natural numbers.
21. Write a program to calculate sum of first N even natural numbers
22. Write a program to calculate sum of squares of first N natural numbers.
23. Write a program to calculate sum of cubes of first N natural numbers.
24. Write a program to calculate x^y (x power y). (x and y are given by user).
25. Write a program to count digits in a given number.
26. Write a program to calculate sum of digits in a given number.
27. Write a program to reverse a number.
28. Write a program to calculate sum of squares of digits in a given number.
29. Write a program to calculate sum of cubes of digits in a given number.
30. Write a program to print all Armstrong numbers under 1000.
31. Write a program to check whether a given number is Prime or not.
32. Write a program to print all Prime numbers between two given numbers.
33. Write a program to print immediate next prime number of a given number.
34. Write a program to print all Prime factors of a given number. (if user enters 36, your output should be 2, 2, 3, 3)
35. Write a program to check whether a given pair of numbers is co-prime or not.
36. Write a program to find LCM of two numbers.
37. Write a program to find HCF of two numbers.
38. Write a program to print decimal number into binary representation
39. Write a program to print ASCII chart.
40. Write a program to print first N terms of Fibonacci series.
41. Write a program to print sum of first N prime numbers.
42. Do all-star pattern problems. (<https://www.mysirg.com/programming-examples/c-programs/>)