To:
Tenouk

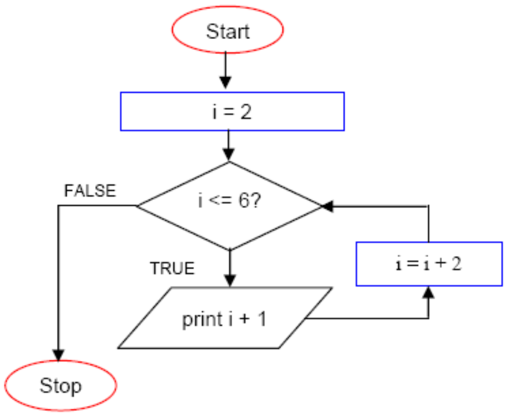
C LAB WORKSHEET 7_1
A C & C++ Repetition: The for Loop 2

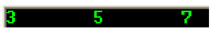
Items in this page:

1. More for loop exercises, questions and answers.
2. The related tutorial reference for this worksheet are: C & C++ program control 1 and C/C++ program control 2.

d. Show the output for the following C code snippets and draw a flowchart for each of them.

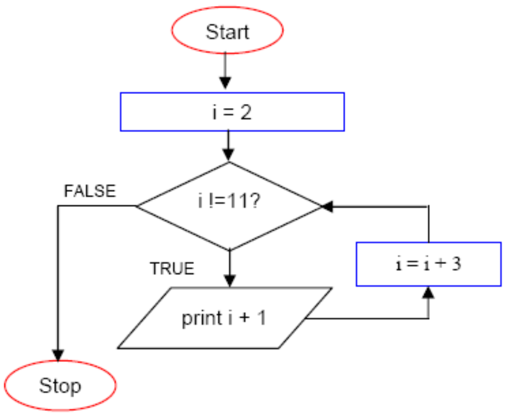
1. for(i = 2; i <= 6; i = i + 2)
   printf("%d\t", i + 1);

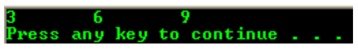




```
3       5       7
```

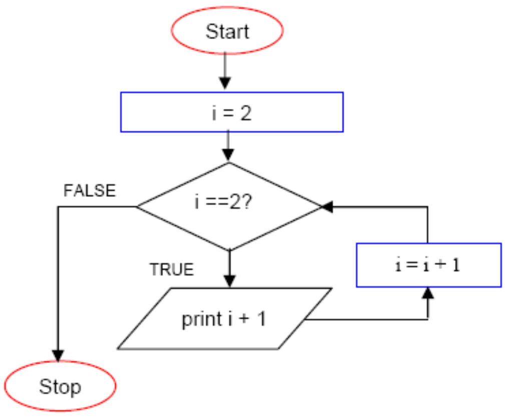
2. for(i = 2; i != 11; i = i + 3)
   printf("%d\t", i + 1);

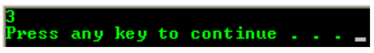




```
3       6       9
Press any key to continue . . .
```

3. for(i = 2; i == 2; i = i + 1)
   printf("%d\t", i + 1);

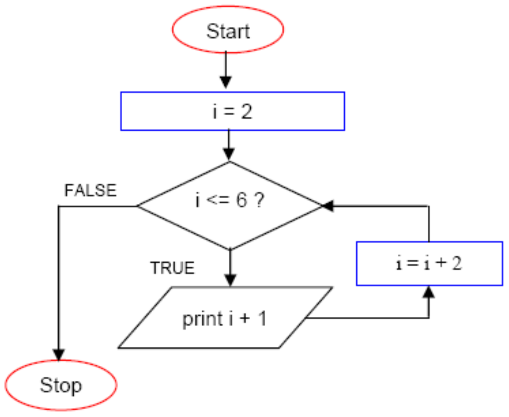


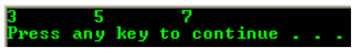


```
3
Press any key to continue . . . _
```

4. for(i = 2; i <= 6; i = i + 2)
printf("%d\t", i + 1);

```
for(i = 2; i <= 6; i = i + 2)
        printf("%d\t", i + 1);
```

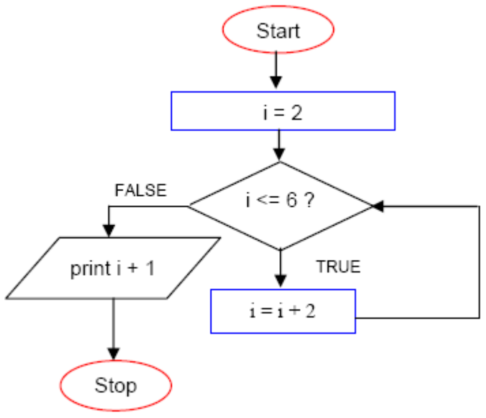


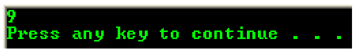


```
3       5       7
Press any key to continue . . .
```

5. for(i = 2; i <= 6; i = i + 2); // note the semicolon here
printf("%d\t", i + 1);

```
for(i = 2; i <= 6; i = i + 2);  // note the semicolon here
        printf("%d\t", i + 1);
```





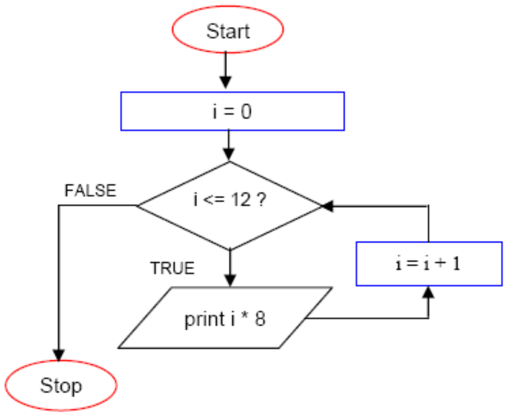
```
9
Press any key to continue . . .
```

e. Print the times table for 8 in one column. To solve this problem you will need to:

1. Write down the pseudocode.
2. Build a flow chart.
3. Create a C program.

The following is a sample pseudocode.

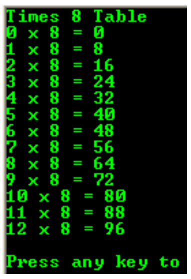
```
Declare a variable of type integer and set the initial value to 0, int i = 0;
For repetition we need to use loop, for loop.
Start the for loop.
Set the terminal condition, i <=12;
Increment i by 1, i = i + 1;
For every iteration, times 8, i * 8;
Print the result with '\n' for every iteration.
Stop the for loop.
```



2.

```
#include <stdio.h>

void main()
{
    int i = 0, j = 8;
    printf("Times 8 Table\n");
    for(i = 0; i <= 12; i = i + 1)
    {
        printf("%d x %d = %d\n", i, j, j*i);
    }
    printf("\n");
}
```

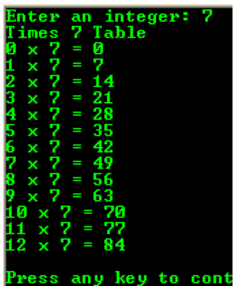
3.



```
Times 8 Table
0 x 8 = 0
1 x 8 = 8
2 x 8 = 16
3 x 8 = 24
4 x 8 = 32
5 x 8 = 40
6 x 8 = 48
7 x 8 = 56
8 x 8 = 64
9 x 8 = 72
10 x 8 = 80
11 x 8 = 88
12 x 8 = 96

Press any key to
```

f. Ask the user for an integer and print out that integer's times table.

```c
#include <stdio.h>

void main()
{
    int i, j;
    printf("Enter an integer: ");
    scanf("%d", &i);
    printf("Times %d Table\n", i);
    for(j = 0; j <= 12; j = j + 1)
    {
        printf("%d x %d = %d\n", j, i, j*i);
    }
    printf("\n");
}
```



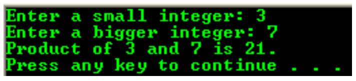
```
Enter an integer: 7
Times 7 Table
0 x 7 = 0
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
11 x 7 = 77
12 x 7 = 84

Press any key to cont
```

g. Ask the user for two integers, first a small one then a larger one. Multiply these two integers by doing repeated addition. For example, if we were to obtain 3 and 5 from the user, then we would find their product by adding the larger one (5) three times.

The following is a pseudocode. This pseudocode has been edited many times.

```
Declare two integer variables to hold the small and bigger integers
        int x = 0, y = 0;
Prompt user to enter an integer,
        printf("Enter an integer: ");
Read and store the integer,
        scanf_s("%d", &x, sizeof(int));
Prompt user another bigger integer,
        printf("Enter another integer bigger than previous one: ");
Read and store the integer in variable y,
        scanf_s("%d", &y, sizeof(int));
Repetition of addition operations, we use for loop.
The for loop starts
        Initial value, i = x, use x value and then decrement it by 1 for
        each iteration
        Terminal condition, i >=1, the last iteration must be 1
        Iteration, a decrement, x = x – 1, decrement by 1 for each
        loop iteration.
        The for loop operation(s), declare and initialize variable
        named sum to hold the sum of the add-up
        For each iteration, add-up the bigger integer,
            sum = sum + y;
The for loop terminates.
Display the sum.
        printf("Product of %d and %d is %d\n", x, y, sum);
```

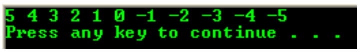
```c
#include <stdio.h>

void main()
{
    int i = 0, x = 0, y = 0, sum = 0;
    printf("Enter a small integer: ");
    scanf_s("%d", &x, sizeof(int));
    printf("Enter a bigger integer: ");
    scanf_s("%d", &y, sizeof(int));
    for(i = x; i >=1; i = i -1)
        sum = sum + y;
    printf("Product of %d and %d is %d\n.\n", x, y, sum);
}
```



```
Enter a small integer: 3
Enter a bigger integer: 7
Product of 3 and 7 is 21.
Press any key to continue . . .
```

h. Print out all integers from 5 down to -5.

```c
/* Print out all integers from 5 down to -5 */
#include <stdio.h>

void main()
{
    int i;
    for(i = 5; i >=-5; i = i-1)
    printf("%d ", i);
    printf("\n");
}
```



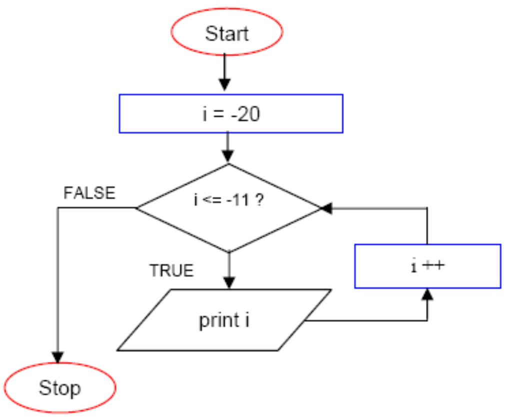
```
5 4 3 2 1 0 -1 -2 -3 -4 -5
Press any key to continue . . .
```

i. Print out all integers from -11 down to -20. Provide the following information for this problem.

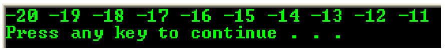
1. Write down the pseudocode.
2. Build a flow chart.
3. Create a C program.

The following is a sample pseudocode.

```
Declare and initialize a variable for iteration,
int i = 0;
The for loop starts.
        Initial value, i = -20
Terminal condition, i <= -11
Iteration, increment, i = i + 1 or i++
Print the value for each iteration, printf("%d ", i);
The for loop stops.
```



```c
/* Print out all integers from -11 down to -20. */
#include <stdio.h>
void main()
{
    int i;
    for(i = -20; i <=-11; i++)
        printf("%d ", i);
    printf("\n");
}
```

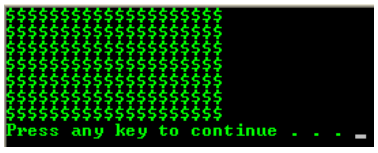


```
-20 -19 -18 -17 -16 -15 -14 -13 -12 -11
Press any key to continue . . .
```

j. Print out $$$$$$ on seven consecutive lines.

The source code is given below and the output is on the right.

```
/* print out $$$$$$ on seven consecutive lines.
    We make more $$$$ in this program! */
#include <stdio.h>

void main()
{
    // used for row and column
    int i, j;
    // row, each row execute internal for loop
    for(i = 1; i <= 7; i++)
    {
        // every column, print '$'
        for(j = 1; j <= 20; j++)
            printf("%c", '$');
        // go to next line
        printf("\n");
    }
}
```



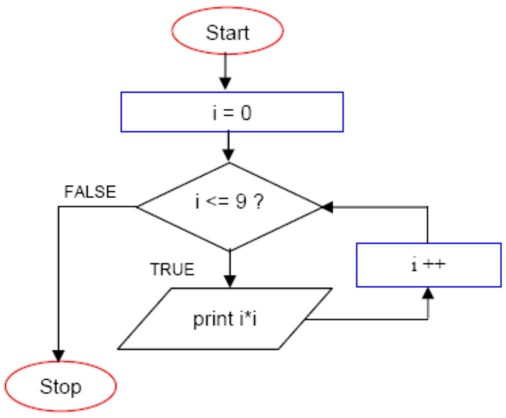
```
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
Press any key to continue . . .
```

k. Print out the squares of the first 10 integers. To solve this problem you will need to:

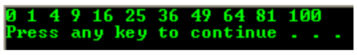
1. Write down the pseudocode.
2. Build a flow chart.
3. Create a C program.

The following is a sample of pseudocode.

```
Declare an initialize a variable to hold an integer.
int i;
The for loop starts.
Initial value, i = 0;
Terminal condition, i <=9.
Iteration, increment, i = i + 1 or i++.
for loop operation,
Square, i*i.
Print the result, printf(“%d “, i*i);
The for loop stops.
```



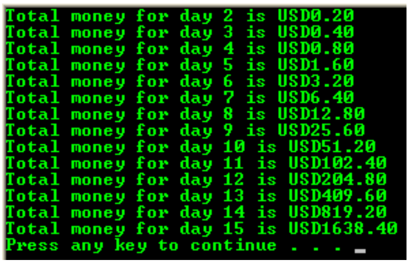
```
/* Print out the squares of the first 10 integers */
#include <stdio.h>

void main()
{
    int i;
    for(i = 0; i <= 10; i++)
        printf("%d ", i*i);
    printf("\n");
}
```



```
0 1 4 9 16 25 36 49 64 81 100
Press any key to continue . . .
```

l. Suppose you are given ten cent on day 1 and on day 2 you are given twice as much. If each day you are given twice as much money as on the previous day, then on day 15, how much money you will receive? Build a C program to find the solution.

```
#include <stdio.h>

void main()
{
    int i;
    // first day got 10 cent
    double sum = 0.10;
    // start on 2nd day, got twice
    for(i = 2; i <= 15;)
    {
        // the next day got twice the previous day
        sum = sum + sum;
        printf("Total money for day %d is USD%.2f\n", i, sum);
        // after complete the calculation, go to the next day
        i++;
    }
}
```

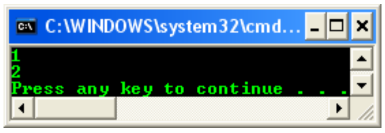


```
Total money for day 2 is USD0.20
Total money for day 3 is USD0.40
Total money for day 4 is USD0.80
Total money for day 5 is USD1.60
Total money for day 6 is USD3.20
Total money for day 7 is USD6.40
Total money for day 8 is USD12.80
Total money for day 9 is USD25.60
Total money for day 10 is USD51.20
Total money for day 11 is USD102.40
Total money for day 12 is USD204.80
Total money for day 13 is USD409.60
Total money for day 14 is USD819.20
Total money for day 15 is USD1638.40
Press any key to continue . . .
```

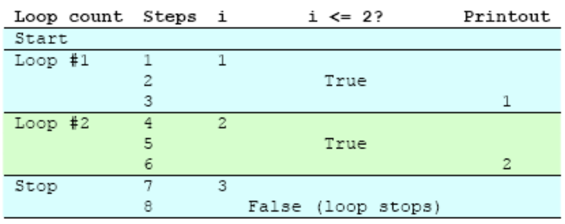
More Example And Practice

- All loops must start somewhere; this called the initialization of the loop. They must stop sometime, the termination of the loop, or else they keep executing, resulting the **infinite loop** (use **CTRL-C** to terminate the program for PC compatible). To terminate a loop, we will need to evaluate conditions, for example, whether a variable is equal to a value or not. Furthermore, while the loop is going through its iterations, it must come closer and closer to the terminal condition.

```
#include <stdio.h>

void main()
{
     int i;
     i = 1;                      // Statement 1
     for(; i <= 2;)              // Statement 2
     {
          printf("%d\n", i);   // Statement 3
          i = i + 1;             // Statement 4
     }
}
```

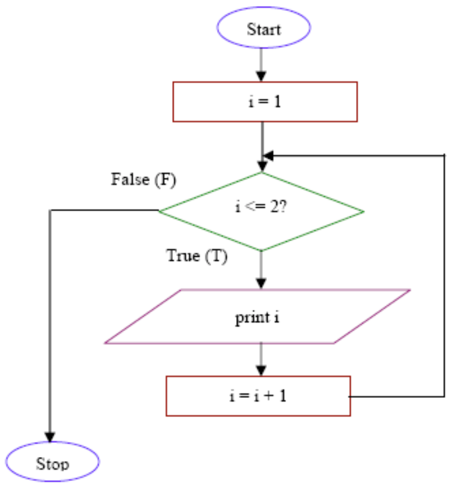


```
1
2
Press any key to continue . . .
```



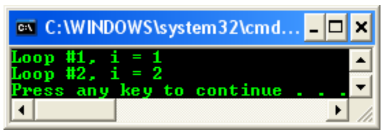
Loop count	Steps	i	i <= 2?	Printout
Start				
Loop #1	1	1		
	2		True	
	3			1
Loop #2	4	2		
	5		True	
	6			2
Stop	7	3		
	8		False (loop stops)	

- The previous code illustrates a very simple loop. The numbers in the comments identify selected statements. A tracechart is shown alongside the code. In step 1 of the trace, statement 1 of the code is executed, that is 1 is assigned to i. In statement 2 a question is asked. Is i less than or equal to 2? Yes, so the loop, which is composed of the two statements inside the set of braces, is executed. Statement 3 shows that i, with a value of 1, is printed and then i is incremented to 2. Here, we draw a horizontal line in the tracechart to depict that the execution goes up to the for statement.
- In step 5, is i less than or equal to 2? Yes, so we go through the loop again, printing 2 for i and incrementing it to 3. Now we have complemented the second iteration of the loop. In step 8, checking to see if i <= 2, we see that it is false and we stop the loop. The flowchart is given below.



- The flowchart starts at the oval labeled Start and ends at the one labeled Stop. A rectangle is used for assignments. i is assigned 1 to begin. Then we encounter a decision diamond. Here we take the T for the true route and fall into the loop until the condition of i <= 2? becomes false, where we stop.
- With a minimum number of changes, convert this flowchart so that all the integers from 3 to 7 are printed and write the code.
- Write the code and draw the flowchart that will print 3, 5 and 7 instead of 3, 4, 5, 6 and 7. From the flowchart write the tracechart.
- You can show the iteration number of the loop by modifying the previous program as shown below.

```
#include <stdio.h>

void main()
{
      int i, count = 0;
      i = 1;
      for(; i <= 2;)
      {
            count = ++count;
            printf("Loop #%d, i = %d\n", count, i);
            i = i + 1;
      }
}
```

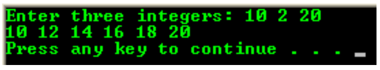


```
Loop #1, i = 1
Loop #2, i = 2
Press any key to continue . . .
```

Write a program that will ask the user to give three integers. Call these integers start, step_by and stop. After these three integers are scanned in, set up the for loop that will start i at the value of start, make it increase by the value given by step_by and make it stop at the value stored in stop. Print these values as shown in the following output sample.

```
-----------Output---------------
Enter three integers: 23 3 32
23 26 29 32
```

```
#include <stdio.h>

void main()
{
    int i, start = 0, step_by = 0, stop =0;
    printf_s("Enter three integers: ");
    scanf_s("%d %d %d", &start, &step_by, &stop);
    for(i = start; i <=stop; i = i + step_by)
        printf("%d ", i);
    printf("\n");
}
```

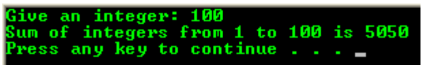


```
Enter three integers: 10 2 20
10 12 14 16 18 20
Press any key to continue . . .
```

Write a program that will add up all the integers from 1 to the integer that was scanned into the variable j. Store the sum in the variable called sum and use i to increment the integers from 1 to j. Print only sum. For example, if 5 were read into j, then sum would be 1 + 2 + 3 + 4 + 5 or 15. A sample output is given below.

```
---------Output----------------
Give an integer: 6
Sum of integers from 1 to 6 is 24.
```

```
#include <stdio.h>

void main()
{
    int i, j, sum = 0;
    printf_s("Give an integer: ");
    scanf_s("%d", &j);
    for(i = 1; i <=j; ++i)
        sum = sum + i;
    printf("Sum of integers from 1 to %d is %d\n", j, sum);
}
```

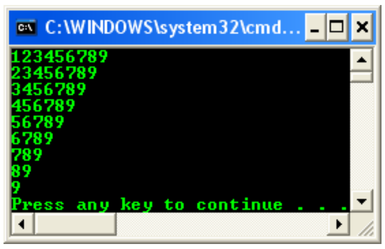


```
Give an integer: 100
Sum of integers from 1 to 100 is 5050
Press any key to continue . . .
```

The Nested Loop: Loop in Loop

- The for loop and all other repetition constructs can also be nested to any degree. In a simple word, nesting means loop in loop. Let try the following program example.

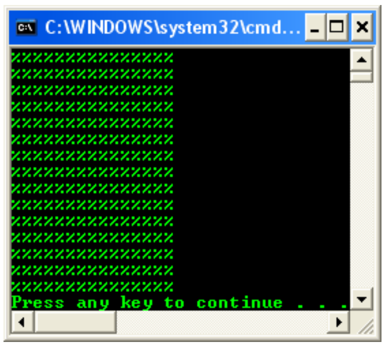
```
// a program to show the nested for loops
#include <stdio.h>

int main()
{
     // variables for counter…
     int i, j;
    // outer loop, execute this first...
    // for every i iteration, execute the inner loop
     for(i=1; i<10;)
     {
        // display i
       printf("%d", i);
       // then, execute inner loop with loop index j the initial value of j is i + 1
       for(j=i+1; j<10; )
       {
            // display result of j iteration
            printf("%d", j);
            // increment j by 1 until j<10
            j = j + 1;
       }
          // go to new line
          printf("\n");
          // increment i by 1, repeat until i<10
          i = i + 1;
     }
     return 0;
}
```



```
123456789
23456789
3456789
456789
56789
6789
789
89
9
Press any key to continue . . .
```

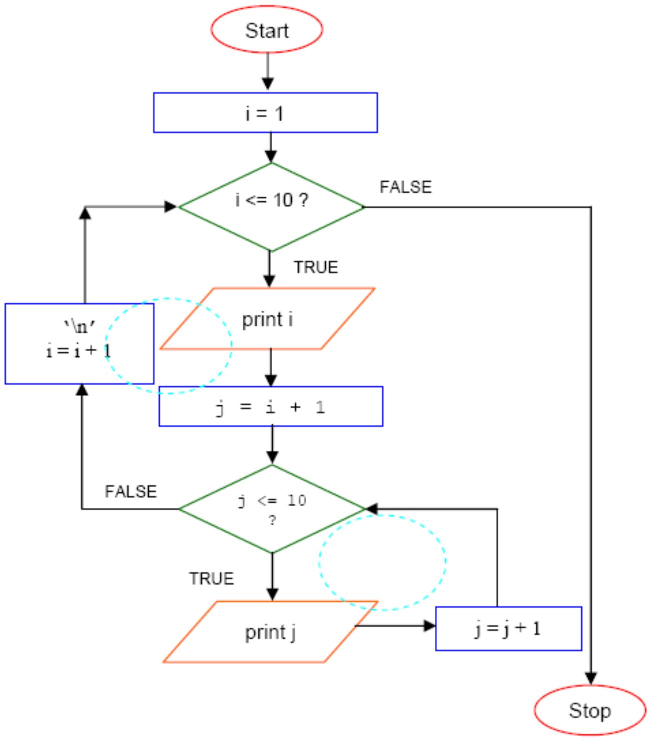
- The program has two for loops. The loop index i for the outer (first) loop runs from 1 to 9 and for each value of i, the loop index j for the inner loop runs from i + 1 to 9. Note that for the last value of i (i.e. 9), the inner loop is not executed at all because the starting value of j is 10 and the expression j < 10 yields the value false because j = 10.
- The following is another nested example; study the program code and the output. In general, a nested two for loops can be depicted as a row and column.

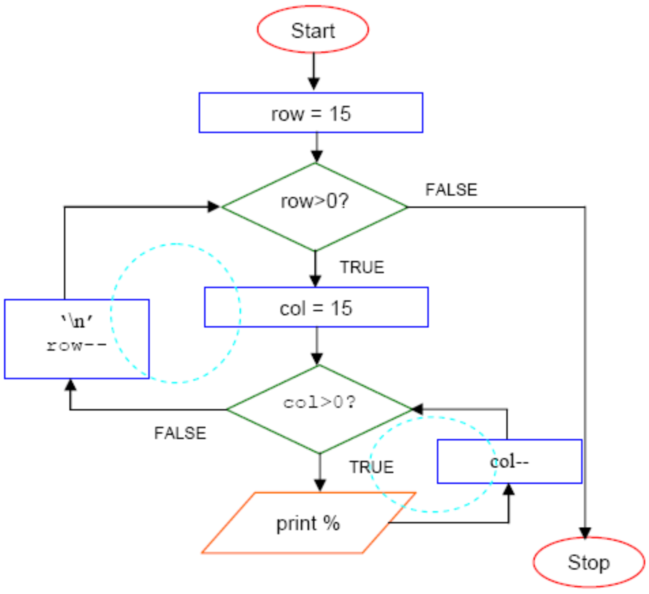
```
/* nesting two for statements – printing row and column */
#include <stdio.h>

void  main()
{
   int  row, col;
   // row, execute outer for loop...start with the preset value and decrement until 1
   for( row = 15; row > 0; row--)
   {
         // column, execute inner loop...start with preset col, decrement until 1
         for(col = 15; col > 0; col--)
         // print %....
         printf("%%");
         // decrement by 1 for inner loop...go to new line for new row...
         printf("\n");
   }
   // decrement by 1 for outer loop, means next row
}
```



```
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
Press any key to continue . . .
```

- The first for loop is executed until the row is 1 (row > 0). For every row value, the inner for loop will be executed until col = 1 (col > 0). In a simple word, the external for loop will print the row and the internal for loop will print the column.
- For the previous two nested for loop program example, build the flowcharts. **Ans:** as shown below.


Start
i = 1
i <= 10 ?
FALSE
TRUE
print i
'\n'
i = i + 1
j = i + 1
j <= 10 ?
FALSE
TRUE
print j
j = j + 1
Stop


Start
row = 15
row>0?
FALSE
TRUE
col = 15
'\n'
row--
col>0?
FALSE
TRUE
col--
print %
Stop