SINDH MADRESSATUL ISLAM UNIVERISTY, KARACHI

DEPARTMENT OF SOFTWARE ENGINEERING

FALL 2022

CSC103 - PROGRAMMING FUNDAMENTALS

ZUBAIR-UDDIN SHAIKH

SECTION SE1A/SE1B/SE1C/CS1D[e]

LAB MANUAL 07

NESTED LOOPS IN C

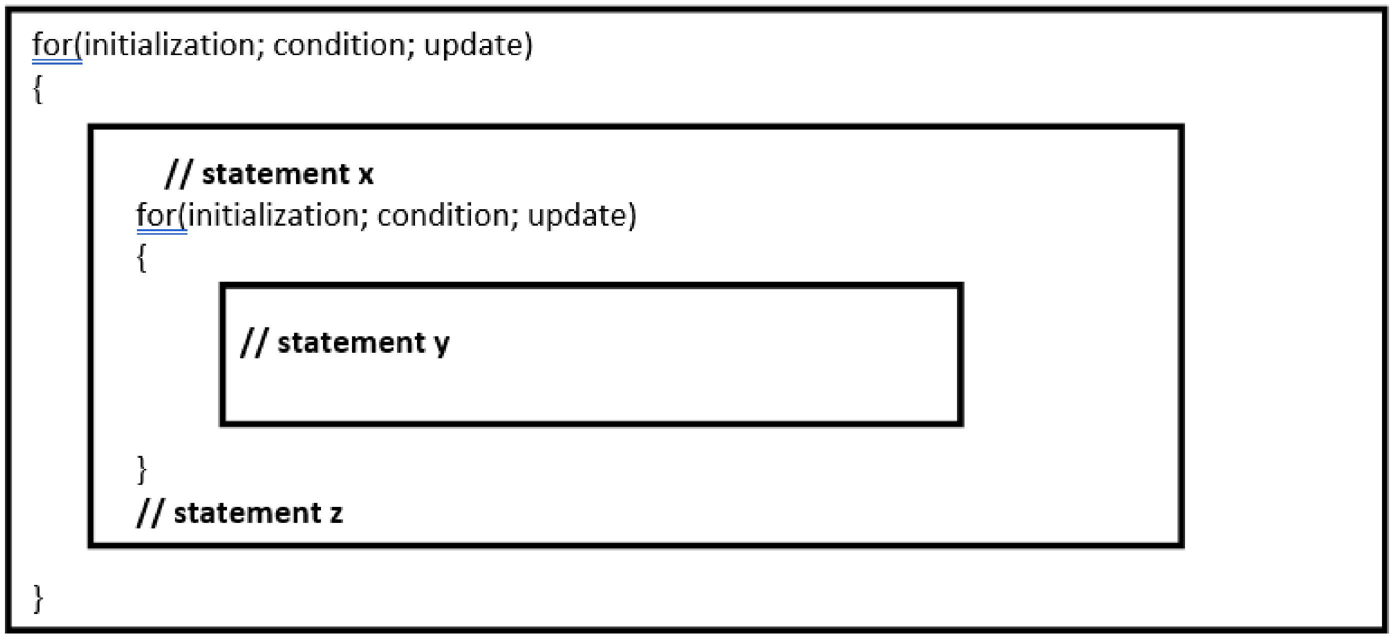
NESTED LOOPS IN C

1. Nested Loops:

- A nested loop is a loop within a loop, an inner loop within the body of an outer one.
- How this works is that the first pass of the outer loop triggers the inner loop, which executes to completion.
- Then the second pass of the outer loop triggers the inner loop again.
- This repeats until the outer loop finishes.

2. Nested "for" Loop:

- A for loop can contain any set of statements in its body/block– even another for loop.
- The inner loop must have a different name for its loop control variable so that it does not conflict with the outer loop.



```
for(initialization; condition; update)
{
       // statement x
      for(initialization; condition; update)
      {
              // statement y
      }
      // statement z
}
```

Example 01: First dry run the following code and write the output on your notebook. Then compile and run and verify your dry run results with the program output!Carefully note when and how the values of i and j are changing

```
for(i=0;i<10;i++)
{
   for(j=10;  j>1;  j--)
   {
        printf("%d\t%d\n", i ,j);
   }
}
```

3. Nested while and do-while loops

- Similar to the nested "for" loop, nesting can also be done using while and do while loops.

Task 01: Convert the for loops of Example 01 into while loops.

Hint: Try to convert one for loop into while. And then move on to nesting. Note that in the while loop you have to explicitly do the initialization and updates unlike a for loop.

Task 02: Convert the for loops of Example 01 into do-while loops.

Hint: Try to convert one for loop into do-while. And then move on to nesting. Note that in the do-while loop you have to explicitly do the initialization and updates unlike a for loop.!

Example 02:** **First dry run the following code and write the output on your notebook. Then compile and run and verify your dry run results with the program output!

```
for(i=0; i<5;i++)

{
   for(j=0;j<5;j++)
   {
        printf("*");
   }
   printf("\n");
}
```

***Example 03:* First dry run the following code and write the output on your notebook. Then compile and run and verify your dry run results with the program output! (Note: ++i and j=j+2)**

```
for(i=0; i<3; ++i)
{
   for(j=0;j<10;j=j+2)
   {
        printf("%d%d",i,  j);
   }
   printf("\n");
}

printf("i=%d  j=%d", i, j);
```

- update part can be any valid C statement not just i++
- i++ is known as the post-increment operator
- ++i is known as the pre-increment operator
- There's no difference in both as far as they're written separately i.e. ++i; or i++;

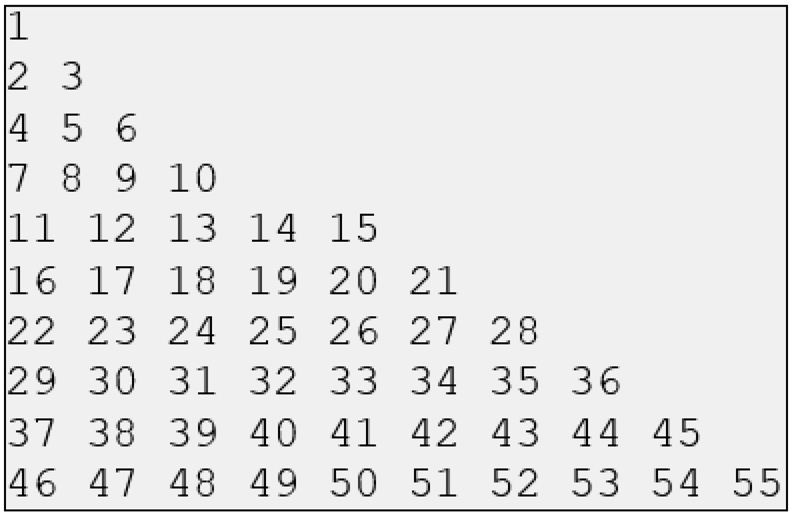
4. Jump statements:

- break (jump to the statement immediately following the current loop or switch statement)
- continue (jump to the condition of the loop)
- Note: The use of jump statements should in general be avoided. They should be used only in specific situations. We will only mention them briefly.

Example 04

```
  for(i=1; i<10; i++)
{
     for(j=0; j<5; j++)
     {
          int product = i*j;
          if(product>15)
              break;
          printf("i=%d, j=%d\n", i, j);
     }
     printf("Hello\n");
}
```

Task 03:Write a c program to print following triangle by using loop



```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55
```