CS1410 : Computer Programming FA17

Lab #05: Arrays, nested Loop Statements

Instructor: Saif Hassan / Muhammad Suffian dated : 22/11/2017

Part A

Objective:

Learning, How to implement nested repetition statements

Learning, How to implement arrays , insertion, deletion & updating in array

Instructions:

- Perform all Examples as shown
- First complete the examples and assignments then try and test your own queries

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.

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 the right. 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);
    }
}
```

DRY RUN OUTPUT

While loop : Concept code

```c
#include <stdio.h>
main()
{
      int count=1;
      while (count <=4)
      {
            printf("%d ", count);
            count++;
      }
}
```

Do-while loop concept code:

```c
#include <stdio.h>

int main () {
   int a = 10;

   do {
        printf("value of a: %d\n", a);
        a = a + 1;
        }while( a < 20 );
   return 0;
}
```

Nested while and do-while loops

Similarly, nesting can also be done using while and do while loops.

Example 02: 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 forloop.

Example 03 : 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 04 : First dry run the following code and write the output on the right. 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");
}
```

DRY RUN OUTPUT

Example 05 : First dry run the following code and write the output on the right. 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);
```

DRY RUN OUTPUT

// 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++;***

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 06:

```
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");
}
```

Example 07:

```
for(i=1; i<10; i++)
{
      for(j=0; j<5; j++)
      {
            int product = i*j;
            if(product<25)
                  continue;
            printf("i=%d, j=%d\n", i, j);
      }
      printf("Hello\n");
}
```

Example 08:

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
```

Part B

1. **(triangle.c)**Write a program that prints the following pattern where **n** is taken as input:

n = 2

```
 *
* *
```

n = 3

```
  *
 * *
* * *
```

n = 4

```
   *
  * *
 * * *
* * * *
```

And so on…

2. **(diamond.c)**Write a program that prints the following pattern where **n** is taken as input:

n = 2

```
 *
*  *
 *
```

n = 3

```
  *
 *  *
*  *  *
 *  *
  *
```

n = 4

```
   *
  *  *
 *  *  *
*  *  *  *
 *  *  *
  *  *
   *
```

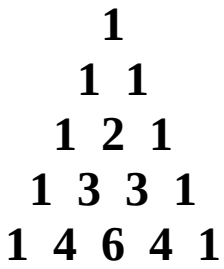
And so on…

3. Write a program that finds a power b (a^b), where a and b are inputs, **without using the multiplication operator *. [Only while loops allowed]**

4. Write a program that prints all perfect number from 1 to 100.

   Hint [**Aninteger number is said to be a perfect number if its factors including 1 (but not the number itself) sum to the number.]**

5. Write a C program to print following number triangle :



Note: To build the triangle, start with "1" at the top, and then continue placing numbers below it in a triangular pattern. Each number is just sum of above two number, (except for the edge, which are all '1' and all numbers outside the Triangle are 0's).

Part C

Array

```
How to Declare the array :
data_type array_name[array_size];
float mark[5];
```

Walkthrough task:

```c
//Program to find the average of n (n < 10) numbers using arrays

#include <stdio.h>
int main()
{
     int marks[10], i, n, sum = 0, average;
     printf("Enter n: ");
     scanf("%d", &n);
     for(i=0; i<n; ++i)
     {
          printf("Enter number%d: ",i+1);
          scanf("%d", &marks[i]);
          sum += marks[i];
     }
     average = sum/n;
```

```
    printf("Average marks = %d", average);

    return 0;
}
```

Exercise1:
Write a C program to read elements in an array from user and count total number of duplicate elements in array.[Hint: nested for loop]

Searching in arrays : concept code

```
/* search num in inputArray from index 0 to elementCount-1 */
   for(counter = 0; counter < elementCount; counter++){
       if(inputArray[counter] == num){
           printf("Number %d found at index %d\n", num, counter);
           break;
       }
   }
```

Exercise2:
Write a C program to read elements in an array and do the linear search whether an element exists in the array or not.

Exercise3:
Write a C program to delete an element from an array in specified position.[Hint:override element].

Exercise4 :
Write a C program to insert an element in array at specified position.[Hint:Move Element forward].

Best Wishes