2017

LOOPS – FOR, WHILE

CSE 180 – COMPUTER PROGRAMMING

S2 EEE - LAB 4

KRISHNA PRASAD | JAN – JUNE 2017

1. Write a C program that prints all the natural numbers from 1 to n, where n is the number entered by the user.

```c
//Program to print numbers from
//1 to n
#include <stdio.h>
int main()
{
    int i,n;
    printf("Enter a number");
    scanf("%d",&n);
    //Using loop
    for(i=0;i<=n;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

Hmm, I kind of didn't correctly understand this

"Ok, lemme explain this, do you recognise the diagram given below. Take a moment and study it."

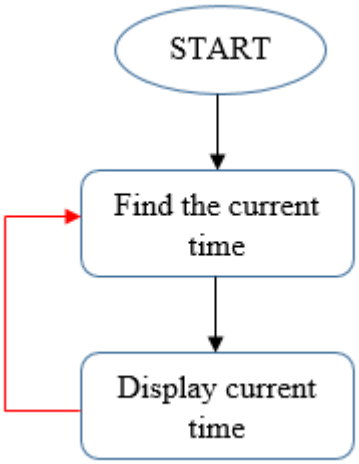
"What does it reveal?"

"There are three steps, isnt it so?"

"Step 1: Get the current time"

"Step 2: Display the current time"

"Step 3: Go back to finding the current time and repeat the procedures from Step 1"



Pic - 1

"In short, it represents a LOOP, in this case a loop that does not have an ending. So, it's an infinite loop."

"If you observe closely, it has a starting point, but it does not have a terminating statement or condition or in other words no condition that makes it stop or no ending point"

Hmm, alright, it's starting to make sense now, but why do we need it?

"Well, if you are asked to print a certain sentence like "OM LOKAH SAMASTHA SUHKHINO BHAVANTHU", say 100 times, how would you do it?

With the knowledge we currently possess we would have to use printf 100 times, isnt it?"

"But with loops, to start with, we can avoid it and reduce the code considerably"

2. Let's try out another program. Let's say as part of a module in a bigger project, you have to find the sum of all the first n numbers. (You are not allowed to use the formula for this)

   "Now, over here, we do not know what the number n will be. It could be any number entered by the user"

```c
Program to find the sum
of the first n numbers
*/
#include <stdio.h>
int main()
{
    int sum=0,i,n;
    printf("Enter a number");
    scanf("%d",&n);
    //Using loop
    for(i=0;i<=n;i++)
    {
        sum = sum + i;
    }
    printf("The sum is %d",sum);
    return 0;
}
```

Ok, when I look at the code, I understand it, but if I was to write it on another occasion, I get stuck.

Let's see, *how did I know that I have to use a <u>FOR LOOP</u>*? I think that's your question isnt it?

```c
/*
Program to find the sum
of the first n numbers
*/
#include <stdio.h>
int main()
{
    int sum=0,i,n;
    printf("Enter a number");
    scanf("%d",&n);
    //Using loop
    for(i=0;i<=n;i++)
    {
        sum = sum + i;
    }
    printf("The sum is %d",sum);
    return 0;
}
```

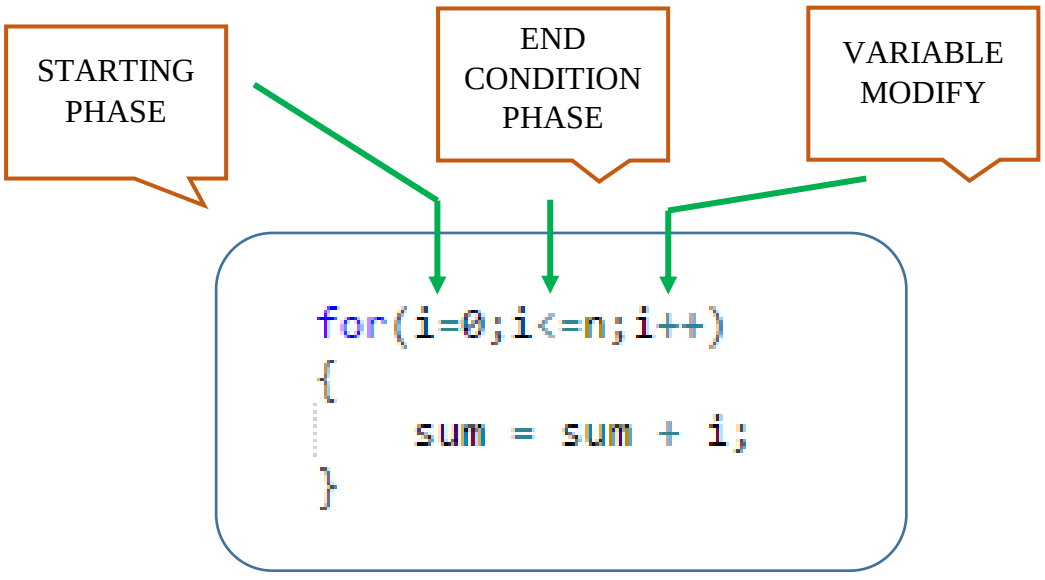
I think you need to focus on the three pillars of FOR LOOP

1. **Starting phase**
2. **End condition**
3. **Variable modify phase**

Filter out the **starting value** of the variable and the **ending value.**


```c
/*
Program to find the sum
of the first n numbers
*/
#include <stdio.h>
int main()
{
    int sum=0,i,n;
    printf("Enter a number");
    scanf("%d",&n);
    //Using loop
    for(i=0;i<=n;i++)
    {
        sum = sum + i;
    }
    printf("The sum is %d",sum);
    return 0;
}
```


Let's magnigy the FOR LOOP from the code given in the left



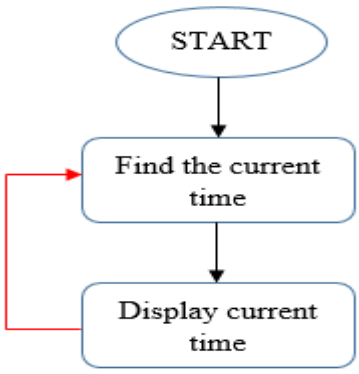
With this information, let's revisit Pic 1 . we see that there

Is a starting condition, but there is no ending condition.

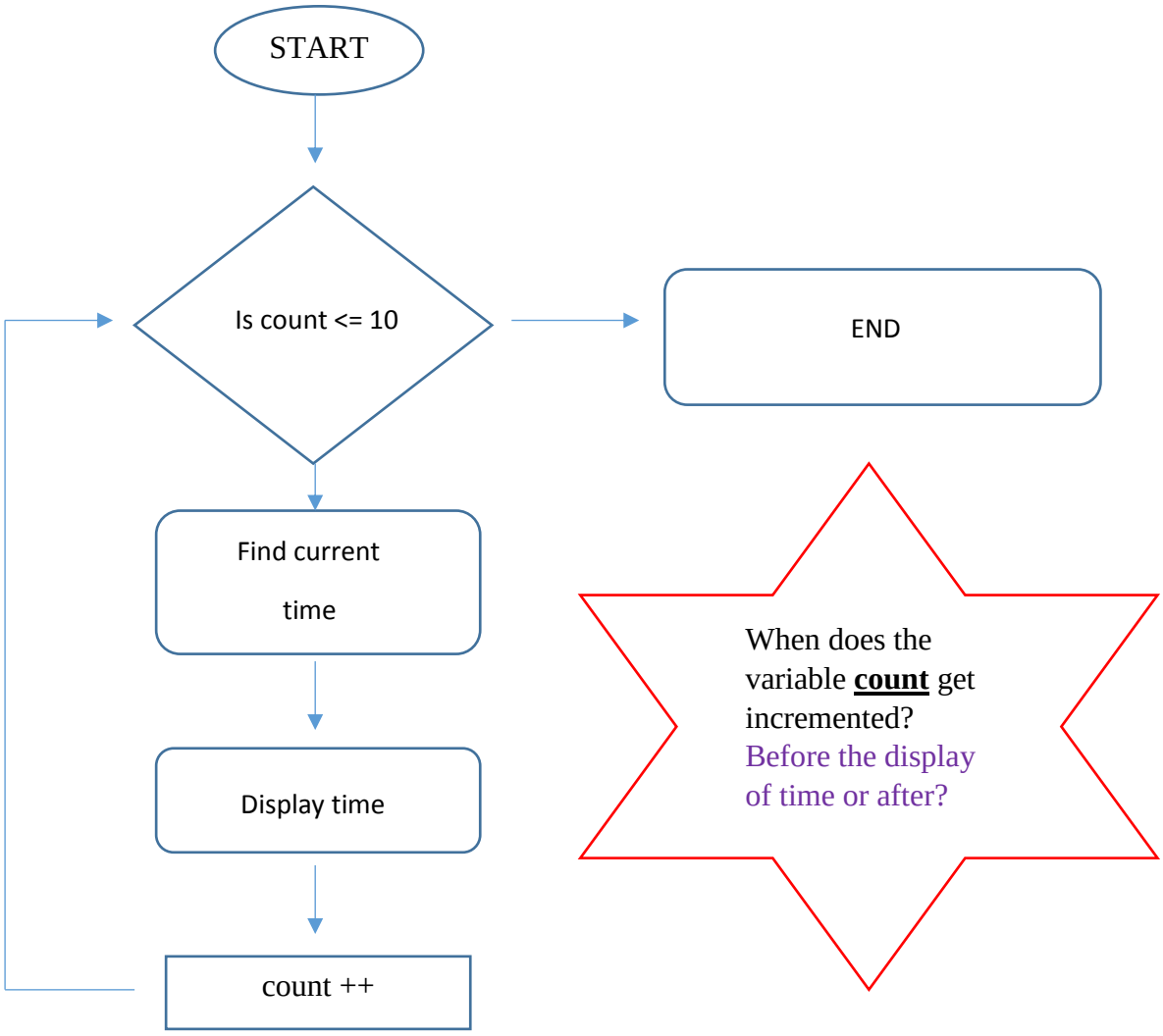
So let's bring in a ending condition,

Count <= 10

Which means that the current time should be displayed only ten times.




```
for(i=0;i<=10;i++)
{
    DISPLAY TIME
}
```



3. Write a program that prints all the numbers from 50 to 500.

Fill in the blanks

```c
#include<stdio.h>
int main()
{
    int i;
    for(i=  ;i<=   ;i++)
    {
        printf("%d",i);
    }
    return 0;
}
```

4. Write a program that prints all the even numbers from 1 to m (m not included).

```c
#include<stdio.h>
int main()
{
    int i,m;
    printf("Enter a number");
    scanf("%d",&m);
    for(i=0;i<m;i++)
    {
        if(m%2==0)
        printf("%d",m);
    }
    return 0;
}
```

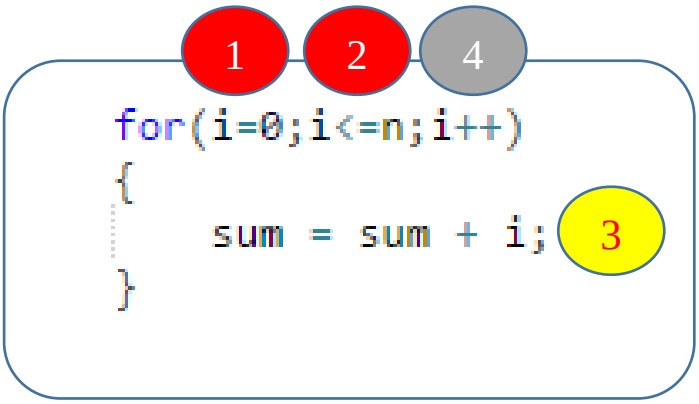
5. Write a program that prints all the odd numbers from 1 to m (m included).
6. Write a program that prints all the numbers from n to 1 (1 included).

```c
#include <stdio.h>
int main()
{
    int n;
    printf("Enter a number");
    scanf("%d",n);
    //Remember the starting value is n
    // Also the terminating value is 1
    for(i=n;i>=1;i--)
    {
        printf("%d",i);
    }
    return 0;
}
```

You have the starting value, n and you have the ending value, which is 1.

Take a moment to study the condition. From **N to 1**. So we have to **decrease** the value of **i from N to 1**

Let's take some time to understand the execution process in a FOR loop, before we proceed



7. Write a program that multiplies all the numbers from N to M (N > M) with the number 10 and displays the results.
8. Find the output for the following program(s):

```c
#include <stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
    {
    }
    printf("%d",i);
    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int i=0;
    for(i=0;i<10;i++)
    {
    }
    printf("%d",i++);
    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int i=0,j=10;
    for(i=0;i<10;i++,j--)
    {
    }
    printf("%d",j);
    return 0;
}
```

9. There are two numbers whose sum or difference or product has to be found. Write a menu based program which asks the user to enter a OR b or C. If the user enters a, then find the sum of two numbers, if the user enters b, then find the difference of the two numbers and if the user enters c, then find the product of two numbers using **switch statement**. (non loop question)

10. Write a program that asks the user to enter a number and the output should be to display the result which is to multiply the number 10 with the number entered by the user. However if the product obtained is larger than 100 then the program must exit or quit.

```c
#include <stdio.h>
int main()
{
    int n,product;
    char ch;
    while(1)
    {
        printf("Enter a number:");
        scanf("%d",&n);
        product = n *10;
        if(product>100)
        //'break' statement quits the loop
        // and comes out of it.
        break;
        else
        printf("%d",product);
    }
    return 0;
}
```

While loop

Explain 'break' statement

- Can **while loop** be used within a **for loop**?

11. Can you do the program in question no 10 using for loop.

```c
#include <stdio.h>
int main()
{
    int n,product,i;
    for(i=0;;i++)
    {
        printf("Enter a number:");
        scanf("%d",&n);
        product = n *10;
        if(product>100)
        //'break' statement quits the loop
        // and comes out of it.
        break;
        else
        printf("%d",product);
    }
    return 0;
}
```

What's different in this for loop?

12. What is the output for the following program?

```c
#include <stdio.h>
int main()
{
  int i;
  for(i=0;i<10;i+=2)
  {
      printf("%d",i);
  }
  return 0;
}
```

Explain the cause of the output?

13. Write a program that prints the values from 1 to n using while loop. Write the **pseudo code** the following program.

```c
#include <stdio.h>
int main()
{
  int i=0,n;
  printf("Enter a number");
  scanf("%d",&n);
  while(i<=n)
  {
      printf("%d",i);
      i++;
  }
  return 0;
}
```

Explain the working of **while loop**

14. Use While loop to print even numbers between 1 to 25.

15. Parts of the program that reads an integer until 900 is encountered. Also it counts the number of positive, negative and zeroes entered by the user. Fill up the rest of the code with adequate comments.

```
#include <stdio.h>
int main()
{
  int pos_count=0,neg_count=0,ze_count=0,n;
  while(1)
  {

     printf("Enter a number");
     scanf("%d",&n);
     if(n==900)
     break;
     if(n>0)
     {
        ■■■■■■■■■■■■■■■■
     }
     else if(■■■)
     {
        ■■■■■■■■■■■■■■■■■
     }
     else if(n==0)
     {
        ■■■■■■■■■■■■■■■■■
     }
  }
  printf("Positive number count is %d\n",pos_count);
  printf("Negative number count is %d\n",neg_count);
  printf("Zero number count is %d\n",ze_count);
  return 0;
}
```

Explain the **break** statement

16. Write a program that prints the square and cube of the first n natural numbers. In this question, instead of giving you the code, certain hints are given which will help you in coding the required program. (please note that you will not be given any such hints in the exam)

- The C library function **double sqrt(x)** returns the square root of **x**.
- This function returns the square root of x. If x is negative, the sqrt function will return a domain error
- the required header for the sqrt function is:math.h
- The C library function **double pow(double x, double y)** returns **x** raised to the power of **y** i.e. x^y.
- double pow(x,y)
- The pow function returns *x* raised to the power of *y*. If x is negative and y is not an integer value, the pow function will return a domain error.
- the required header for the sqrt function is:math.h

17. Using for loop, write a program that prints the multiplication table of a number. Fill up the incomplete code given.

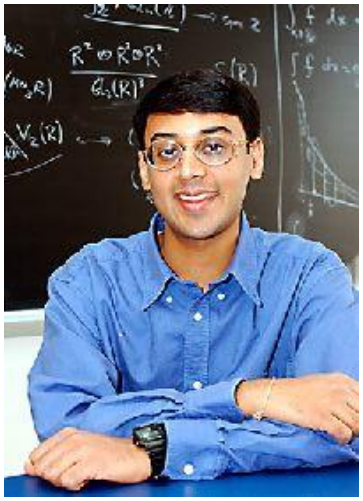
```
#include <stdio.h>
int main()
{
    int n, i;
    printf("Enter an integer: ");
    scanf("%d",&n);
    for(i=1; i<=10; ++i)
    {
        printf("%d * %d = %d \n",[redacted]);
    }
    return 0;
}
```

A sample output

```
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

18. Write a progam to print the factorial of a number.

    In mathematics, the **factorial** of a non-negative integer *n*, denoted by *n*!, is the product of all positive integers less than or equal to *n*. For eg: 3! = 1*2*3. The value of 0! is 1. The factorial operation is encountered in many areas of mathematics, notably in combinatorics, algebra, and mathematical analysis. Its most basic occurrence is the fact that there are *n*! ways to arrange *n* distinct objects into a sequence (i.e., permutations of the set of objects). This fact was known at least as early as the 12th century, to Indian scholars. The most recent contribution in the field of Factorials, by an Indian is that of the Fields medal mathematician who developed the Bhargava factorial as part of his thesis in Harvard University in 1996.



Dr Manjul Bhargava

19. Write a program that calculates the pow(x,y), x^y using loop.
20. Write a program to sum the series **1+1/2+1/3+…+1/n**.
21. Write a program to generate the following ***series*** .You may limit the input with an integer variable.
    a. 2, 4, 16, 256, 65536….

Example:
Sample Input : 500
Sample Output : 2 ,4, 16, 256

22. Write a program to find the number of ***vowels*** in a given string (Read one character at a time using getchar() example given below) . Print the number of each individual vowels and total number of consonants. Write the program using else-if. while ((c = getchar()) != EOF) { //This will read on character at time }// and read till the input is end of file (ctrl+d). EOF stands for **End-of-file** and is a term typically used when tranmission of data ends prematurely. A sample program is given below. Write the **pseudo code** for the entire program.

```
#include<stdio.h>
int main()
{
    char c;
    int vowel_count=0,consonant_count=0;
    while(c=getchar()!=EOF)
    {
        if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u')
        {
            vowel_count++;
        }
        else
        {
            consonant_count++;
        }
    }
}
```

23. Write a program which prints out the character set. Use a ***for statement to loop from 0 to 127***. The body of the **for** loop should be a **printf()** statement which prints out the integer controlling the loop. Print this integer twice, once as a decimal (%**d** format) and once as a character (**%c** format). The output will give you a list of the characters and their corresponding codes. Some of the characters which are not printable but many of them will be recognizable.
24. Write a program which implements a trivial encryption scheme. The program should *read a single character* using getchar(), *add the integer value 13* to the character, and *output the result.* Continue to loop until *getchar()returns EOF*. Also write a corresponding decryption program to restore the input to its original state.
25. Write a ***menu driven program*** to read ***three numbers*** and display the following menu that offers six options

    1. Calculate total.
    2. Calculate average.
    3. Display the smallest.
    4. Display the largest value.
    5. Exit.

    Use *switch case* to check the selection and do the task. The program should work in a loop and quit only when the user selects the *Exit* option.

26. Study the program given below to find the decimal equivalent of a binary number. Also give a **flowchart** and a description of the program. Also write the **pseudo code** for the program.

```
#include <stdio.h>
void main()
{
    int  num, binary_val, decimal_val = 0, base = 1, rem;
    printf("Enter a binary number:");
    scanf("%d", &num);
    binary_val = num;
    while (num > 0)
    {
        rem = num % 10;
        num = num / 10 ;
        decimal_val = decimal_val + rem * base;
        base = base * 2;
    }
    printf("The Binary number is = %d \n", binary_val);
    printf("Its decimal equivalent is = %d \n", decimal_val);
}
```

27. Write a program that converts an octal number into it's corresponding decimal equivalent.

Any octal number can be converted into decimal number. The method is quite similar to the transformation of any binary number to decimal number, the only difference is that in this case the 2 s will be replaced by 8 and everything else i.e. the methods will remain same. So it is quite clear if we want to change any octal number into decimal number we have to start multiplying the digits of the number from right hand side with increasing powers of 8 staring from 0 and finally summing up all the products. The method will be clearer with an example. Suppose we want to convert 123 into decimal number, so we have to follow the procedure as shown below $123_8 = 1\times8^2 + 2\times8^1 + 3\times8^0 = 64 + 16 + 3 = 73$ So, the decimal equivalent of the number 123_8 is 73_{10}

An octal to decimal mapping table is given below for your reference.

Decimal Number	Octal Number
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
10	12
15	17
16	20

28. Write a program to print the **reverse of a decimal number**.
29. Write a program to check whether the entered **number is a palindrome**.
30. Write a program to enter a number and then calculate **the sum of the digits** of a number.