ME 172
Introduction to Computer Programming Language Sessional

Lecture 3: Switch case, break and introduction to loop

Dr. Md Mamun
Professor
Mantaka Taimullah
Lecturer
Dept of Mechanical Engg, BUET

Class Performance Test

- Write a program that will take two integers as input and then find the largest even number between the two.

(i.e. if both the input numbers are even, it will compare and print the larger. If one is odd and one is even, it will print the even. If both are odd, it will print an error message.)

Time: 5 minutes.

switch statement

General form:

```
switch (variable name)
  {
case value#1:  statements;
               break;
.
.
.
case value#n:  statements;
               break;
default:       statements;
  }
```

This value must be a constant value or expression.

REMEMBER THE COLON OPERATOR!!!

- When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through* to subsequent cases until a break is reached.

A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No **break** is needed in the default case.

Example on switch statement -1

This is a program that takes an input number ranging from 1 to 4 and tells you which number is taken as input. There are 4 possibilities and if the input is not within the range the program can identify it.

```
#include <stdio.h>
int main()
{
 int num;
printf("Enter any integer between 1 to 4:");
scanf("%d",&num);
switch(num)
  {
        case 1:  printf("ONE");
                 break;
        case 2:  printf("TWO");
                 break;
        case 3:  printf("THREE");
                 break;
        case 4:  printf("FOUR");
                 break;
        default: printf("OUT OF BOUND");
  }
return 0;
}
```

Variable name

Case value #

```
int main()
{
char letter;
printf("Enter a, b or c");
scanf("%c", &letter);
switch(letter)
  {
        case 'a':  printf("First letter");
                   break;
        case 'b':  printf("Second letter");
                   break;
        case 'c':  printf("Third letter");
                   break;
        default: printf("OUT OF BOUND");
  }
}
```

Example Problem

- Write a program that will take an expression containing two numbers and an arithmetic operator between them (e.g. 2+3 or 8/4) from keyboard and will print out the result. (Use switch)

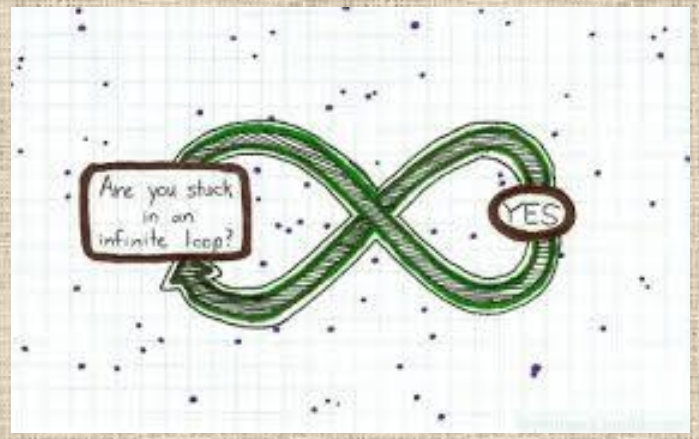
- If + is entered, it will add the two numbers,
  if – is entered, it will subtract the two numbers…….
  Make sure that an error will be printed should 0 be given as a divisor.

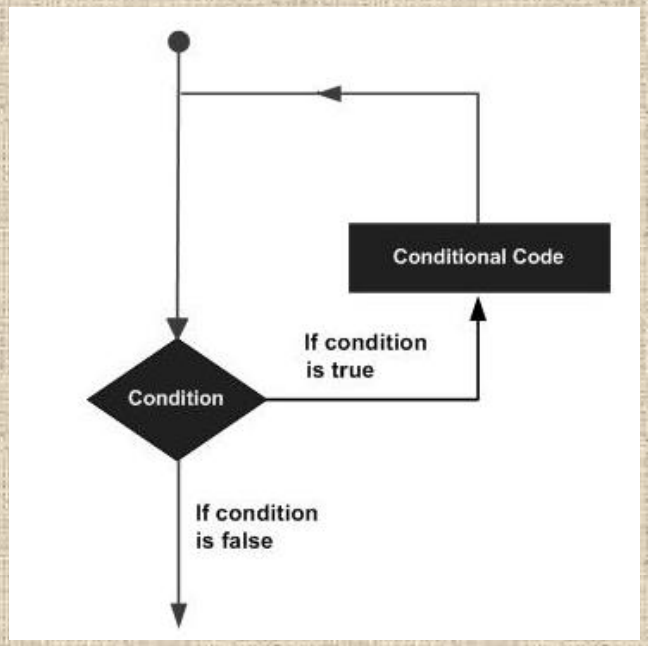
SOLUTION

```
#include<stdio.h>
int main()
{
int a,b;
char op;

printf("Enter the expression: ");
scanf("%d %c %d",&a,&op,&b);
```

```
switch(op)
  {
        case '+': printf(" = %d",a+b); break;
        case '-': printf(" = %d ",a-b); break;
        case 'x':
        case '*': printf(" = %d",a*b); break;
        case '/':
              if(b!=0) printf(" = %d",a/b);
              else printf("Value of divisor can't be zero");
              break;
        default : printf("Unknown Operator");
  }
return 0;
}
```

LOOPS!





- You may encounter situations, when a block of code needs to be executed several number of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- Programming languages provide various control structures that allow for more complicated execution paths.
- A loop statement allows us to execute a statement or group of statements multiple times

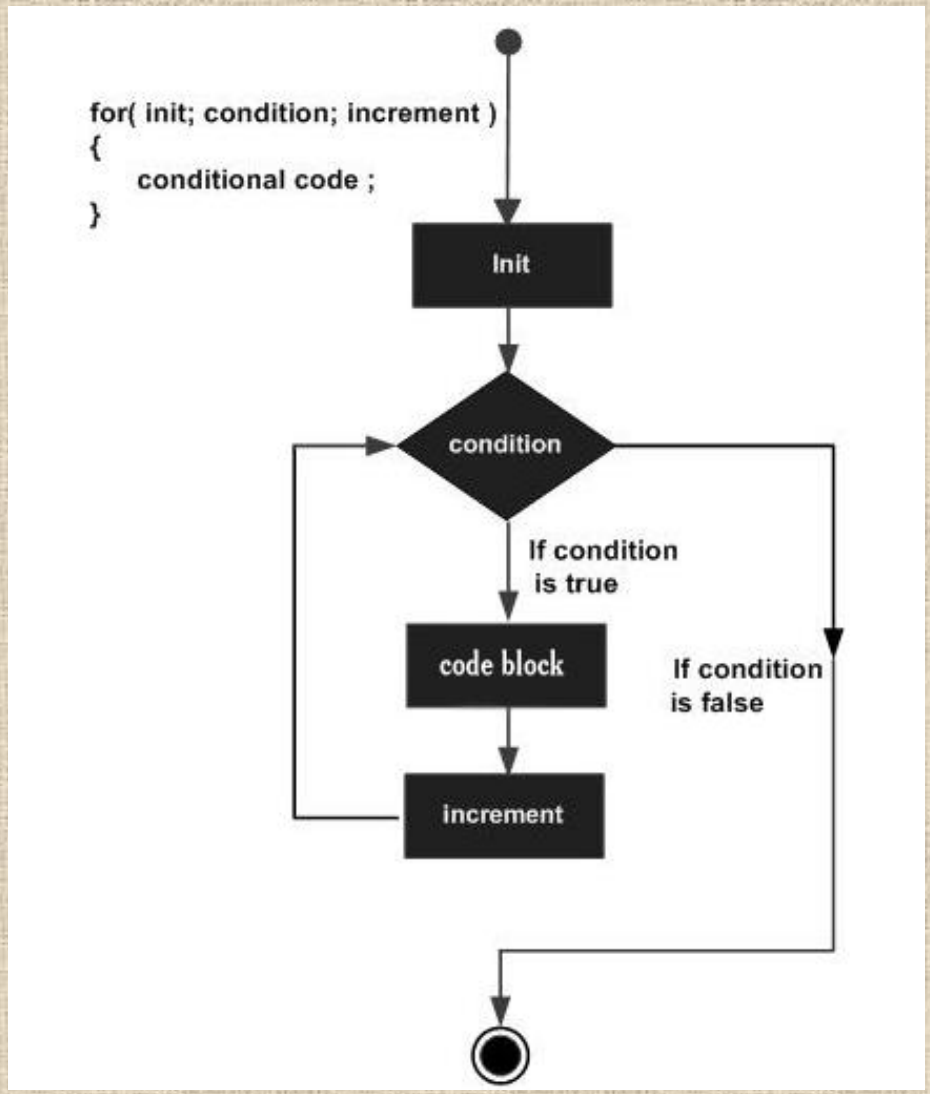
for *loop*

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

General form

for loop

```
for( initialization; conditional test ; increment)
{
 statements;
  ---------------
}
```



Practice Example

- The following is a program that will print the answer of the following series,
  Sum = 1 + 2 + 3 + 4 + …….. + n
  n is a positive integer that will be taken as input from the user.

```
#include<stdio.h>
int main()
{
        int  i, n, sum = 0;
        printf("Enter the value of n: ");
        scanf("%d",&n);

        for(i=1; i<=n; i++){
                sum = sum+i;
        }
        printf("The sum of the series is = %d ", sum);
        return 0;
}
```

Class Performance Test

- Write a program to calculate the factorial of a given positive integer. The input number should be taken from the user through keyboard.

  Use *for* loop.

Answer:

```
#include<stdio.h>
int main(){
 int num, i;
 long fact = 1;

 printf("Enter the value to find the factorial: ");
 scanf("%d",&num);
 for(i=1; i<=num; i++) fact = fact*i;

 printf("Factorial of %d is = %ld ",num, fact);
 return 0;
}
```

Another Example of for loop

- Write a program to evaluate the following series

  $y = x + x^2/2 + x^3/3 + \ldots\ldots\ldots\ldots$nth term

  Use *for* loop.

```
#include<stdio.h>
#include<math.h>
int main()
{
    int i,j,n;
    float x, sum=0;
    printf("Enter x & n: ");
    scanf("%f %d", &x, &n);
    for(i=1;i<=n;i++){
        sum=sum+ pow(x,i)/i;
    }
    printf("%.3f",sum);
    return 0;
}
```

Multiple conditions

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

Commas separate the conditions

```
1       10
2       9
3       8
4       7
5       6
6       5
7       4
8       3
9       2
10      1
```

Nested Loops

```
for ( init; condition; increment )

{       for ( init; condition; increment )

                { statement(s); }

        statement(s);
}
```

C programming allows to use one loop inside another loop.

Example (Nested Loop)

Using *for* loop

```
int main(){
int i, n, line;
printf("How many lines? : ");
scanf("%d", &n);
  for(line=n;line>=1;line--){
     for(i=1;i<=line;i++) {
          printf("%d ",i);
     }
     printf("\n");
   }
return 0;
}
```

Make the following graph

1 2 3 4

1 2 3

1 2

1

Class Performance Test

- Write a program that will take a number, n as input and print a rectangle that will contain n number of * on one side and n+2 number of * on the other side

  For example, if n = 3

  Desired output:

  *****

  *****

  *****

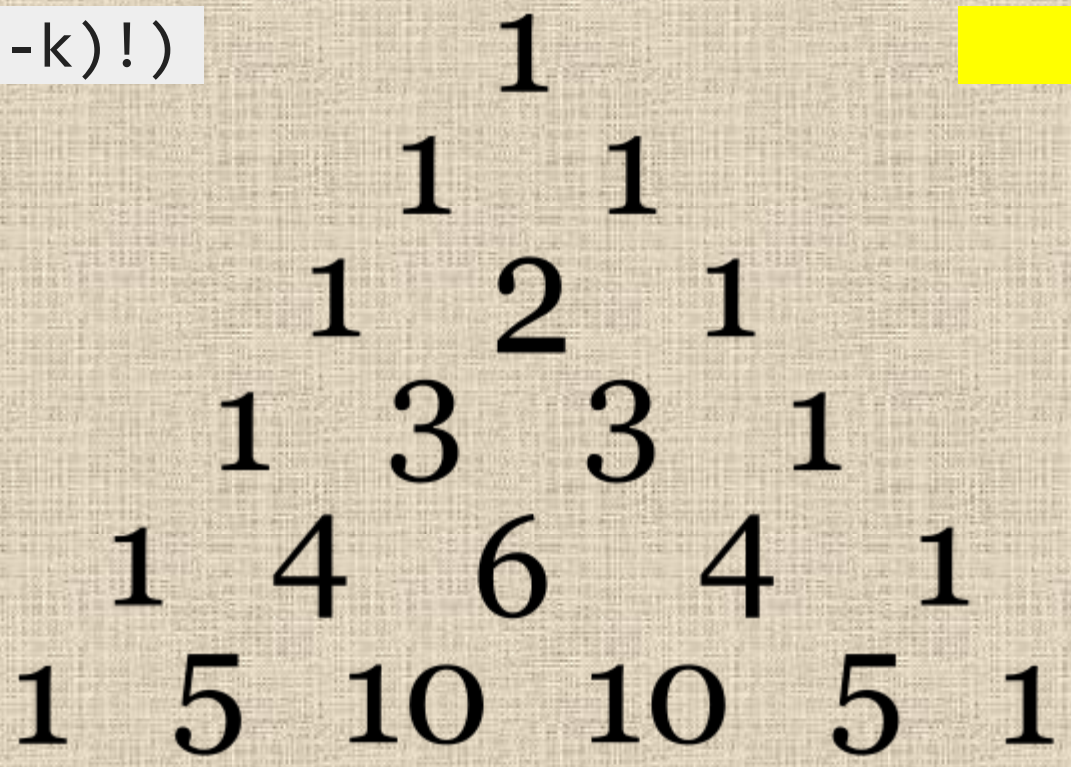
Solution

```
#include<stdio.h>
int main()
{
    int i,j,n;
    printf("Enter the magic number: ");
    scanf("%d",&n);
    for(i=1;i<=n;i++){
        for(j=1;j<=n+2;j++) printf("*");
        printf("\n");
    }
    return 0;
}
```

#Problem 6

Draw a Pascal's triangle like the following one using C programming (for loop)

```
c(n, k) = n! / (k!(n-k)!)
```



Try this at home!

Example problem

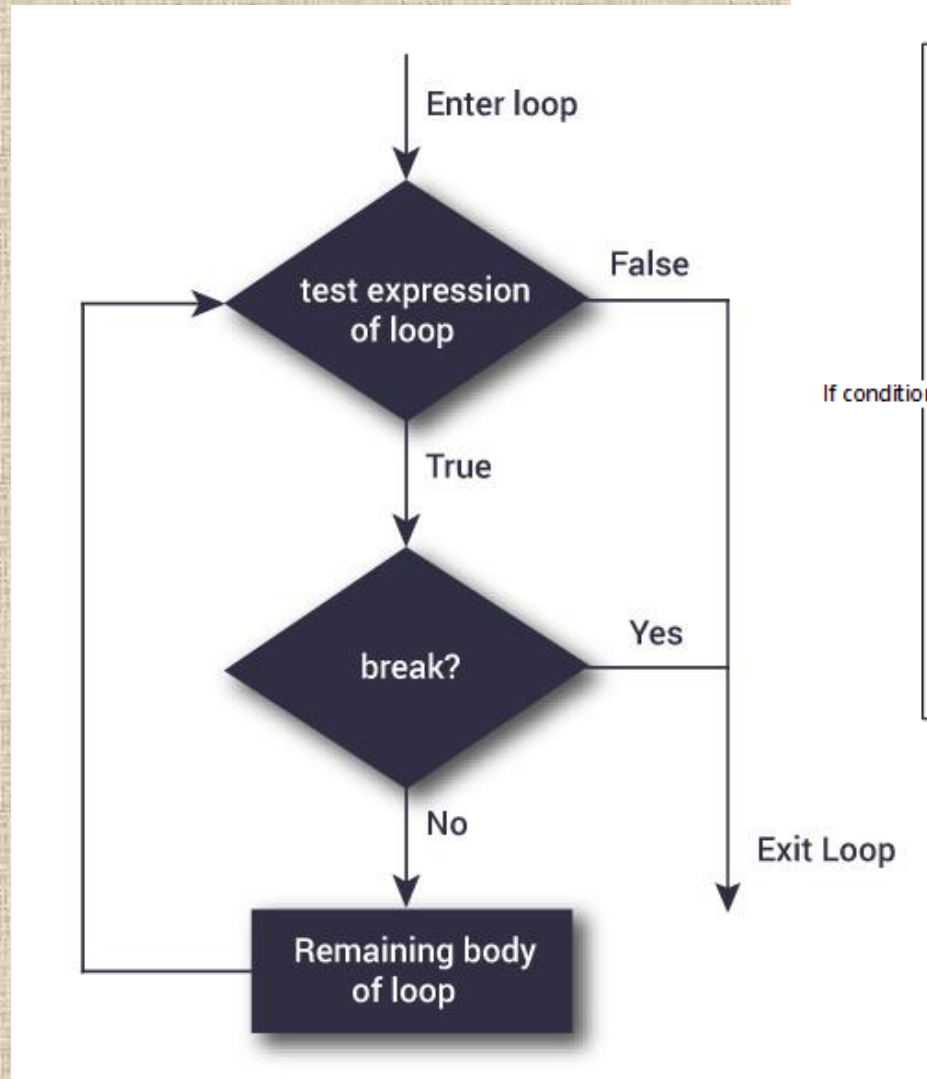
Write down a program to check whether any integer bigger than 1 is a prime number or not

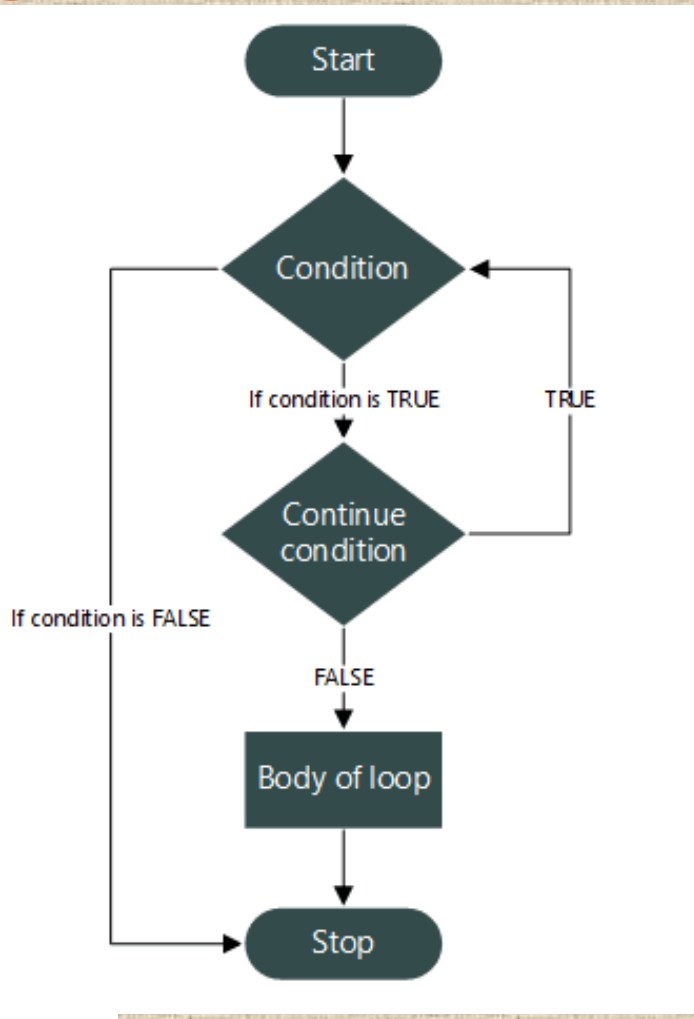
```
#include<stdio.h>
int main()
{
   int i,n, is_prime=1;
   printf("Enter an integer: ");
   scanf("%d", &n);
   for(i=2;i<n;i++){
      if(n%i==0) is_prime=0;
   }
   if(is_prime==1) printf("Prime");
   else printf("Not prime");

   return 0;
}
```

break and *continue* statements

break statements are used to break a loop before reaching the terminating condition. When program finds a ***break*** statement, immediately it jumps to the end of the loop.

continue statements are used to avoid execution of subsequent instructions in a code from a certain point. If it is used inside a loop, the compiler will not execute commands following ***continue*** statement and restart the loop. When program finds a ***continue*** statement, immediately it jumps to the beginning of the loop.





```c
// Program to calculate sum of the positive numbers of 10 input numbers.

# include <stdio.h>
int main()
{
  int i;
  float number, sum = 0.0;
  for(i=1; i <= 10; i++)
  {
    printf("Enter number #%d: ",i);
    scanf("%f",&number);
    if(number < 0.0)
    {
      continue;
    }
    sum += number; // sum = sum + number;
}
printf("Sum = %.2lf",sum);
return 0;
}
```

Recap: Prime number

Write down a program to check whether any integer bigger than 1 is a prime number or not.
Try to make the program efficient by reducing computational effort.

```
#include<stdio.h>
#include<math.h>
int main()
{
    int i,n, is_prime=1;
    printf("Enter an integer: ");
    scanf("%d", &n);
    for(i=2;i<=sqrt(n);i++){
        if(n%i==0){ is_prime=0; break;}
    }
    if(is_prime==1) printf("Prime");
    else printf("Not prime");

    return 0;
}
```

Assignments

1) Write a program to evaluate the *sine* series using *for* loop.

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \ldots\ldots\ldots\ldots 10\text{th term}$$

2) Write a program that determines the number of trailing zeros at the end of X! (X factorial), where X is an arbitrary number. For instance, 5! is 120, so it has one trailing zero.

3) Solve the problem of Pascal's triangle mentioned in the lecture.