ME 172

Introduction to Computer Programming Language Sessional

Lecture 3: Switch case, break and introduction to loop

Saif Al-Afsan Shamim

Lecturer, Dept. of Me, BUET

Courtesy: Dr. Noor Al-Quddus

Dr. Monjur Morshed

M Jamil Hossain

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

```
#include <stdio.h>
void main(void)
{
 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");
  }
 }
```

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.

Variable name

Case value #

Class Performance Test 1

- Write a program that will take two numbers and an option for arithmetic operation 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 if 0 be given as a divisor.

- Time: 07 minutes!!

SOLUTION

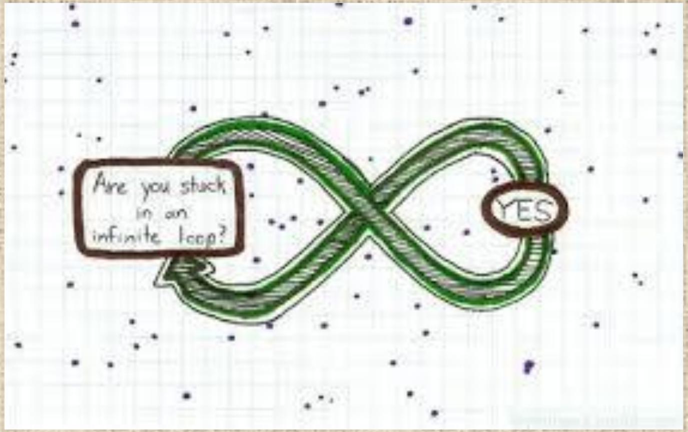
```
#include<stdio.h>
void main(void)
{
int a,b;
char op;

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

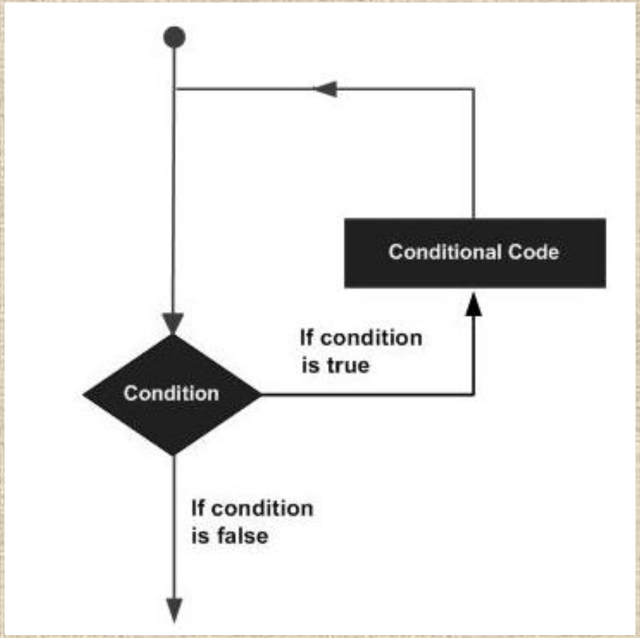
continued to the next page……..

……..continued from the previous page

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

   }

}
```

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

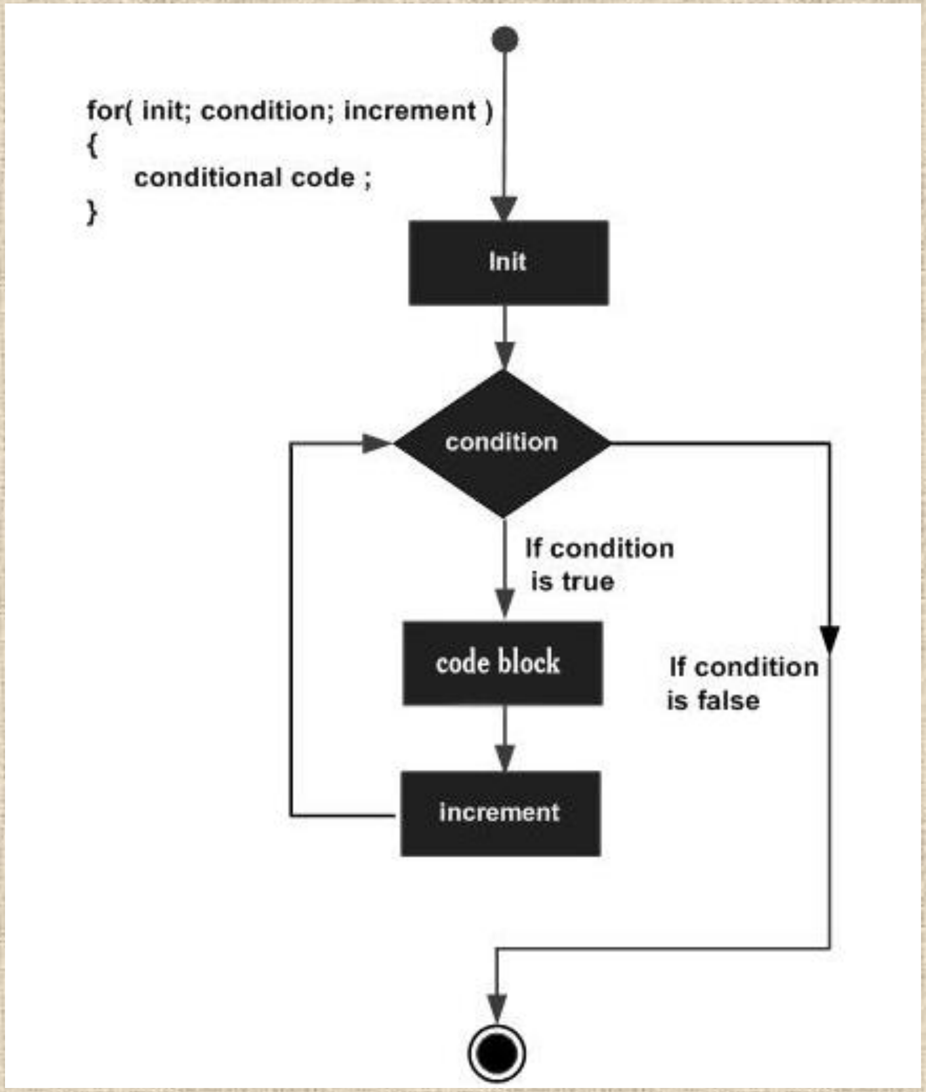
for *loop*

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.

```
void main()
{
        int  i, n, sum = 1;
        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);
}
```

Class Performance Test 2

- 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.

Time: 7 Minutes

Answer:

```
   void main()
{
   int num, i;
   long fact = 1;

   printf("Enter the value to find the factorial: ");
   scanf("%d",&num);

   for(i=num; i>=1; i - -)
          fact = fact*i;
   printf("Factorial of %d is = %ld ",num, fact);
}
```

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 initializations.
But you need to use logical operators for conditions..

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 n,line,i;
printf("\nHow many line:=");
  scanf("%ld",&n);
  for(line=n;line>=1;line--)
    {
      for(i=1;i<=line;i++)
        {
         printf(" %d",i);
         }
     printf("\n");
    }
```

Make the following graph

```
1 2 3 4
1 2 3
1 2
1
```

Example of for loop

- Write a program to evaluate the following series

  $y = x + x^2/2 + x^3/3 + \ldots\ldots\ldots\ldots.$15th term

  Use *for* loop.

```
#include<math.h>

void main(){
   int i,x;
   float y=0.0;
   printf(“Enter x:”);
   scanf("%d",&x);

   for ( i=1;i<=15; i++ ) {
            y +=  pow(x,i) / i ;
     }
   printf("Result: %f",y);
}
```

Class Performance Test 3

- 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>
#include <stdlib.h>

int main()
{
    int n,j,k;
    printf("Enter the magic number: ");
    scanf("%d",&n);
    for(j=1;j<=n;j++)
    {
        for(k=1;k<=n+2;k++)
            {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!

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

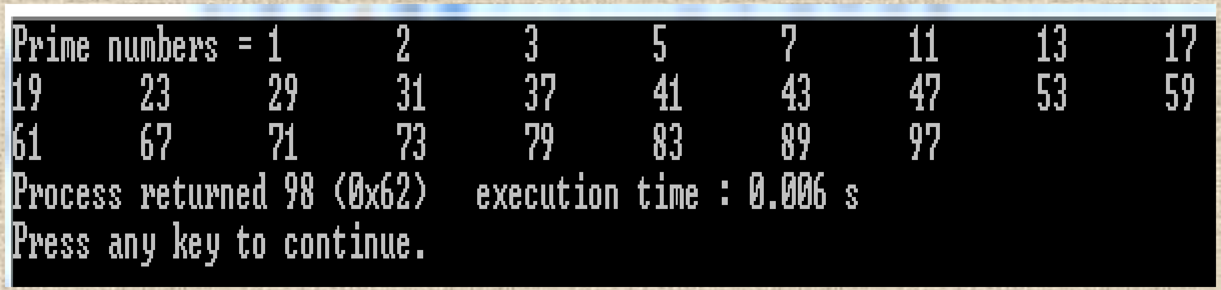
Class Performance test 4

- Write a program to find the prime numbers between 0 to 100.

  Use nested *for* loops.

Time: 15 Minutes!

```
void main()
{
   int i,j, is_prime;
   printf("Prime numbers = ");
   for (i=1; i<100; i++)
         {
            is_prime = 1;
            for (j=2; j<=i-1; j++)
               {
                          if ( (i%j) == 0)
                                      is_prime = 0;
                    }
            if (is_prime == 1)
                         printf("%d\t",i);
         }
}
```

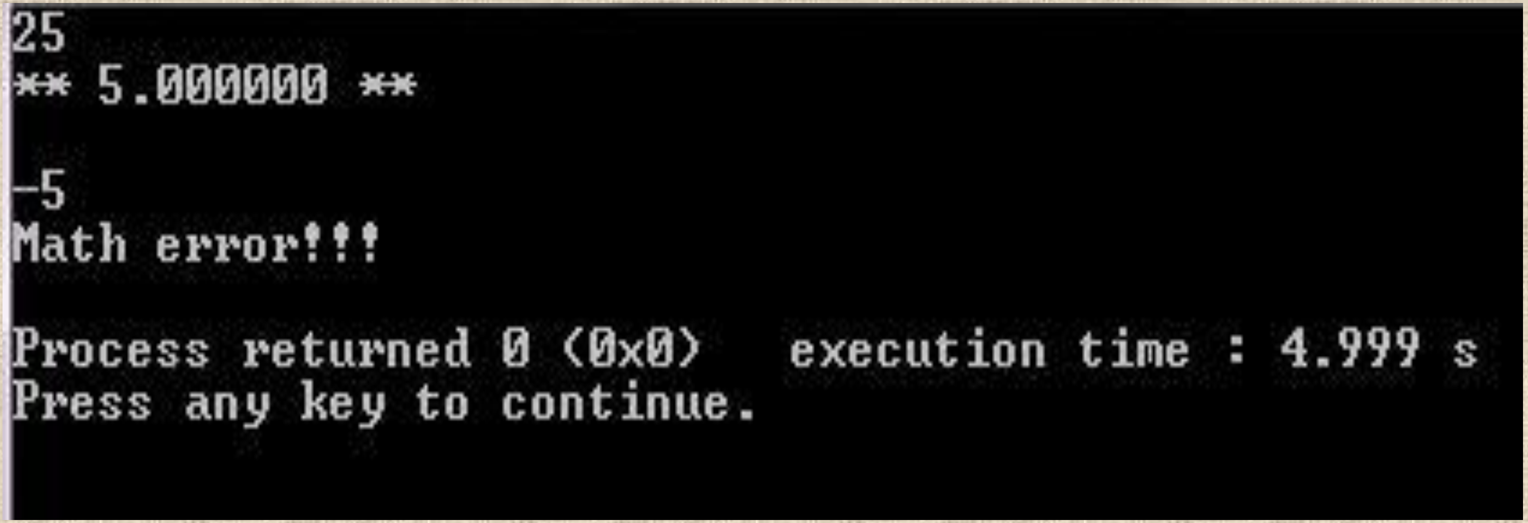


```
Prime numbers = 1       2       3       5       7       11      13      17
19      23      29      31      37      41      43      47      53      59
61      67      71      73      79      83      89      97
Process returned 98 (0x62)   execution time : 0.006 s
Press any key to continue.
```

break and *continue* statements

break statements are used to break a loop before reaching the terminating condition.

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
void main()
{
    float n;
    while(1)
    {
        scanf("%f",&n);
        if(n<0.0) {printf("Math error!!!\n"); break;}
        printf("** %f **\n\n",sqrt(n));
    }
}
```

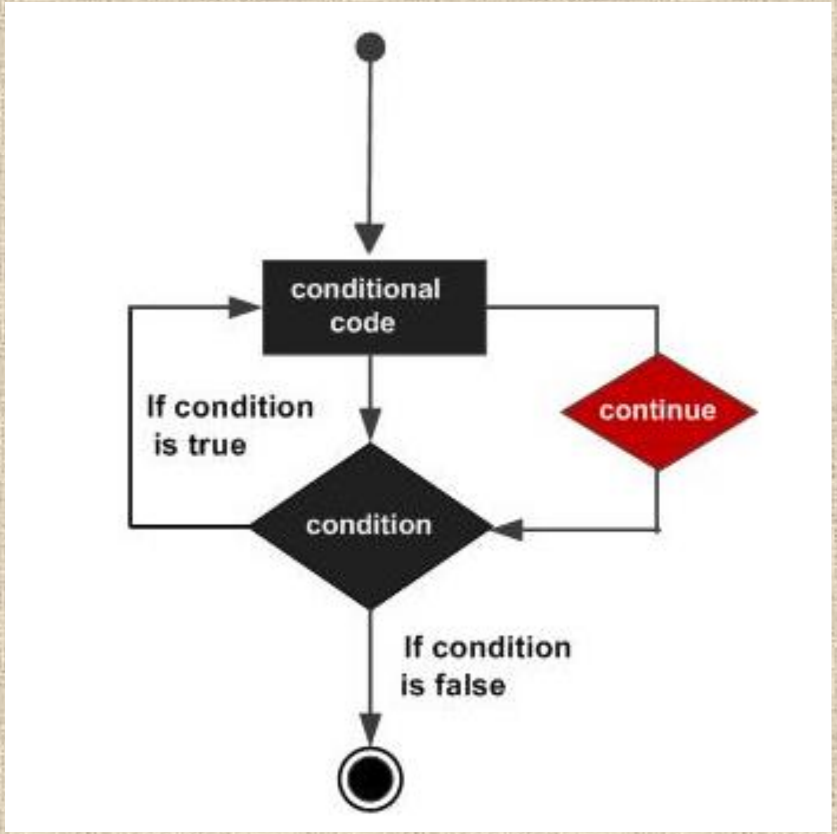


```
25
** 5.000000 **

-5
Math error!!!

Process returned 0 (0x0)   execution time : 4.999 s
Press any key to continue.
```

Output

break **and** ***continue*** **statements**

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.



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.