CHAPTER - 5
LOOPS

CHAPTER 5

LOOPS

LOOP **(Reading)**
WHILE LOOP
FOR LOOP
COMPARING WHILE-LOOP AND FOR-LOOP
INFINITE LOOP
LOOP EXITS
DO WHILE REPETITION
SWITCH STATEMENT
NESTED LOOPS
HOW TO DIVIDE A PROGRAM BETWEEN SEVERAL FILES **(Reading)**

LOOPS

LOOP

- ***Loop*** **allows instructions or statements to be repeated until a *loop* condition is reached to fail.**
- **Some *loops* check the condition at the beginning of the *loop* while others check the condition to repeat again.**

definite repetitions:

- **Definite *loops* check *loop* condition before or after executing statements in the *loop* body.**

indefinite repetitions:

- **Indefinite *loops* do not check the condition for *loop* execution. They are intentional with a purpose.**
- **Exit conditions with in the body of the *loop* reach a condition.**

LOOPS

WHILE LOOP

- *while loop* checks the condition before it allows to execute the statements in the loop body.
- If the condition is false from the start the block of code in the loop body will never be executed.
- Some initial expressions set the condition to be true.

syntax:

```
initialization expressions;
while (condition expressions)
{
    body of statements;
   iterative expressions;
 }
```

```
int main(void)
{
    int count = 1;      // initialization expression
    while (count <= 100)    // condition expression
    {
        printf ("\n", count); //statement
        printf ("%d\n", count); //
        count += 1; // iteration expression
     }
    return 0;
}
```

LOOPS

WHILE LOOP EXAMPLE

```
#include <stdio.h>
int main()
{
   int a;
   a = 0;
   while (a <= 100)
   {
      printf ("%4d degrees F = %4d degrees C\n",
                                    a, (a - 32) * 5 / 9);
      a = a + 10;
   }
   return 0;
}
```

Output of the program:

```
  0 degrees F =  -17 degrees C
 10 degrees F =  -12 degrees C
 20 degrees F =   -6 degrees C
 30 degrees F =   -1 degrees C
 40 degrees F =    4 degrees C
 50 degrees F =   10 degrees C
 60 degrees F =   15 degrees C
 70 degrees F =   21 degrees C
 80 degrees F =   26 degrees C
 90 degrees F =   32 degrees C
100 degrees F =   37 degrees C
```

LOOPS

FOR LOOP

- **For loops initialize variables, check the condition before executing the loop body and execute iterative statements after executing the body of the loop.**
- **For loop is similar to while loop with a different syntax.**

for loop syntax:

```
for (initialization expressions; condition expressions; iterative  expressions;)
{
        body of statements;
}
```

LOOPS

VARIETIES OF FOR LOOP

Counter controlled for loop:

The *for* loop uses a counter variable *count.*

```
for (count = 1; count <= 10; count++)
{
        printf ("%d\n", count);
}
```

multiple initializations:

```
for (i = 0,  j = 100; j != i; i++, j--)
{
     printf ("i = %d,  j = %d\n", i, j);
}
printf ("i = %d,  j = %d\n", i, j);
```

compounding the condition:

```
for (count = 1; (response != 'N') && (count <= 100); count++)
{
        printf ("%d\n", count);
        printf ("Hey man, you still want to continue? (Y/N): \n");
        scanf ("%c", &response);
}
```

LOOPS

SIMPLE FOR LOOP

```
#include <stdio.h>
void main ()
{
    int count = 0;
    printf ("\n**********************");
    for (count = 1 ; count <= 12 ; ++count)
        printf ("\n*                    *");
    printf ("\n**********************\n");
}
```

Output of the program:

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

LOOPS

ANOTHER FOR LOOP

```
#include <stdio.h>
void main ()
{
      int multof;
      int upto = 12;
      printf ("\n Please enter a number to print the \n");
      printf (" multiplication table for (1 to 20): ");
      scanf  ("%d", &multof);
      printf ("\n          Multiplication Table of %d \n\n",
      multof);
      for (int i = 1; i <= upto; i++)
      {
           printf ("\n   %2d  x  %2d   =   %3d ", multof, i,  i
      * multof );
      }
      return;
} /**        end of main program   */
```

```
Output of the program:
Please enter a number (1 to 20)
to print the multiplication
table for: 16
Multiplication Table of 16
16   x    1    =      16
16   x    2    =      32
16   x    3    =      48
16   x    4    =      64
16   x    5    =      80
16   x    6    =      96
16   x    7    =     112
16   x    8    =     128
16   x    9    =     144
16   x   10    =     160
16   x   11    =     176
16   x   12    =     192
```

LOOPS

COMPARING WHILE-LOOP AND FOR-LOOP

- **Difference between while loop and for loop is place of initialization and iteration to alter the condition.**
- **Properly coded while loop and for loop can be interchanged with each other.**

```
INITIALIZATION
while (CONDITION)
{
    BODY
    ITERATION
}
```

```
for (INITIALIZATION; CONDITION;  ITERATION)
{
    BODY
}
```

LOOPS

INFINITE LOOPS

- **Infinite loops are purposely written.**
- **Infinite loops execute without checking the loop condition.**
- **Loop breaks with a loop exit within the body of the loop.**
- **Break, continue and exit are loop exits.**

```
while (true) // condition never fails
{
    Body of statements;
    Condition to break the loop;
}
```

```
for (; ; ;)   // run forever
{
        Body of statements;
        Condition to stop the loop execution;
}
```

LOOPS

INFINITE LOOP EXAMPLE

```
#include <stdio.h>
int main (void)
{
    char ch = 'Q', throwaway = ' ';
    for ( ; ; )
    {
        printf ("\n\n Please enter a letter as your response for
                the action\n");
       printf ("Load, Save, Edit, Quit? : ");
       scanf ("%c%c", &ch, &throwaway);
       if ( ch != 'Q' )
          printf ("\n Your response was %c \n", ch);
       else
       {
                printf ("\n Your response was %c, and
                                exiting\n", ch);
                break;
       }
   } /* end of for loop */
   return 0;
}
```

Output of the above program:

```
Please enter a letter as your response for the action
Load, Save, Edit, Quit? : L
 Your response was L

Please enter a letter as your response for the action
Load, Save, Edit, Quit? : S
 Your response was S

Please enter a letter as your response for the action
Load, Save, Edit, Quit? : E
 Your response was E

Please enter a letter as your response for the action
Load, Save, Edit, Quit? : Q
 Your response was Q, and exiting
```

LOOPS

LOOP EXITS EXAMPLE

- **Loop exits can be used in infinite or continuous loops.**
- **Loop exits are *break*, *continue* and *exit*.**
- **break will break out of the immediate loop.**
- ***continue* will stop executing the remaining statements of the loop body.**
- ***exit* will get out of the program all together.**
- ***goto* will take the flow of program to a designated label**

LOOPS

LOOP EXITS EXAMPLE

```
int main (void)
{
      char gender;
      while (gender = getche ())
      {
          gender = toupper (gender);
          if (gender != 'M' && gender != 'F' )
          {
                printf ("Incorrect gender, please type again\n");
                continue;
          }
          break;
}
```

LOOPS

DO-WHILE REPETITION

- **Do-while loops execute the loop body at least once.**
- **Unlike while loop, Do-while repetitions check loop condition after executing the loop body statements.**
- **Loop is executed until the loop condition false.**

```
syntax:
initialization expressions;
do                    // no condition expressions
{
     body of statements;
     altering expressions;
} while (condition expressions);
```

LOOPS

DO-WHILE LOOP EXAMPLES

```
#include <stdio.h>
int main (void)
{
int value, r_digit;
      printf ("Enter a number to be
     reversed.\n");
      scanf ("%d", &value);
      do
      {
            r_digit = value % 10;
            printf ("%d", r_digit);
            value = value / 10;
      } while (value != 0);
      printf ("%d", r_digit);
     return 0;
}
```

```
unsigned int counter = 5;
unsigned long factorial = 1;
Do
{
        factorial *= counter--; /*Multiply,
      then decrement.*/
} while (counter > 0);
printf ("%lu\n", factorial);
```

LOOPS

SWITCH STATEMENT

- ***switch*** **statement work like** ***if-else*** **statement with variation.**
- ***switch*** **evaluates an integer expression program flow takes path of appropriate** ***case*** **statement.**
- **A** ***break*** **is necessary to break away after executing the statements under the case statement. Otherwise, after executing the statements under the** ***case*****, the program flow checks the next** ***case*** **statement.**
- ***default*** **is required to take the path when the flow could not go into any of the** ***case*** **statements.**
- **No curly braces are required under** ***case*** **statement for multiple statements.**

LOOPS

SWITCH STATEMENT

```
syntax:
switch (int-expression)
{
        constant1: statement case -set 1;
            break;
        case constant2: statement-set 2;
            break;
             ..
             ..
             ..
        case constant N: statement-set N;
            break;
        default: default-statements;
}
```

LOOPS

SWITCH STATEMENT EXAMPLE

```
#include <stdio.h>
int main ()
{
      int grade;
      int aCount = 0, bCount = 0, cCount = 0,
      dCount = 0,  fCount = 0;
      while ( (grade = getchar () ) != EOF)
      {
            switch (grade)
            {
                  case 'A':
                  case 'a':
                        ++aCount;
                        break;
                  case 'B':
                  case 'b':
                        ++bCount;
                        break;
                  case 'C':
                  case 'c':
                        ++cCount;
                        break;
                  case 'D':
                  case 'd':
                        ++dCount;
                        break;
                  case 'F':
                  case 'f':
                        ++fCount;
                        break;
```

LOOPS

SWITCH STATEMENT EXAMPLE

```
                default:
                    printf ("Incorrect letter
                            grade entered.\n");
                    printf ("Enter a new
                            letter grade. \n");
                    break;
            }
        }
        printf ("\n Totals for each letter
                                grade are: \ ");
        printf ("A grade: %d \n", aCount);
        printf ("B grade: %d \n", bCount);
        printf ("C grade: %d \n", cCount);
        printf ("D grade: %d \n", dCount);
        printf ("F grade: %d \n", fCount);
        return 0;
}
```

The results of the output are:

Enter the letter grades.
Enter the EOF character to end input.
A
B
C
C
A
D
F
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
B
Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1

LOOPS

NESTED LOOPS

```
void main ()
{
      long sum = 0L;
      int i = 1, j = 1;     /* loop control variable     */
      int count = 10; /* # sums to be calculated */
      printf ("\t\t number                total \n"); // Output
      for ( i = 1 ;  i <= count ;  i++ )
      {
              sum = 0L;  // Initialize sum for inner loop
              /* Calculate sum of integers from 1 to i */
              for (j = 1 ; j <= i ; j++ )
                  sum += j;
              printf ("\n\t\t%d\t\t\t%ld\n", i, sum);
      }
      printf ("\n\n");
      return;
}
```

Output of the above program:

```
number                total
**********************
1                     1
2                     3
3                     6
4                     10
5                     15
6                     21
7                     28
8                     36
9                     45
10                    55
**********************
```

LOOPS

THREE NESTED LOOPS

```
int main (void)
{
      int i, j, k;

      for (i = 0; i < 3; i++)
      {
           printf ("\n**************************\n");
          for (j = 0; j < 26; )
          {
               for ( k = 0; k < 8 && j < 26; k++)
                    printf ("%c  ",  'A' + j++);
                   printf ("\n");
          }
         printf ("\n**************************\n");
       }
      return 0;
}
```

Output of the above program:

```
**************************
A  B  C  D  E  F  G  H
I  J  K  L  M  N  O  P
Q  R  S  T  U  V  W  X
Y  Z
**************************
**************************
A  B  C  D  E  F  G  H
I  J  K  L  M  N  O  P
Q  R  S  T  U  V  W  X
Y  Z
**************************
**************************
A  B  C  D  E  F  G  H
I  J  K  L  M  N  O  P
Q  R  S  T  U  V  W  X
Y  Z
**************************
```