Loop Control Structures in C

Ex: What is meant by looping? Describe two different forms of looping.

OR

What the different loop control statements used in C? Give the syntax of each.

Solution:

Looping meant, directs a program to perform a set of operations again and again until a specified condition is achieved. This condition causes the termination of the loop. Programming language C contains three statements for looping:

- The *while* loop
- The *do…while* loop
- The *for* loop

while loop: while loop construct contains the condition first. If the condition is satisfied, the control executes the statements following the while loop else, it ignores these statements. The general form of while loop is:

```
while(condition)
{
    statement1;
    statement2;
    ….
}
```

do-while loop: do-while loop construct is another method used in C programming. do-while loop ensures that the program is executed atleast once and checks whether the condition at the end of the do-while loop is true or false. As long as the test condition is true, the statements will be repeated. The control will come out from the loop, only when the test condition is false.

The do-while loop has the following form:

```
do
{
        statement1;
        statement2;
        ………..
}
while(condition);
```

The blocks of statements with in double braces { } following the word do are executed at least once. Then the condition is evaluated. If the condition is true, the block of statements are executed again until the value of condition tested is false.

The for loop statement:

for loop construct is used to execute a set of statements for a given number of times. Thus, it is a shorthand method for executing statements in a loop.

The syntax is:

```
for(initial condition; test condition; incrementer or decrementer)
{
    statement1;
    statement2;
}
```

for loop construct requires to specify three characteristics. These are:

a. The initial value of the loop counter;
b. Testing the loop counter value to determine whether its current value has reached the number of repetitions desired.
c. Increasing or decreasing the value of loop counter by a specified number, each time the program segment is executed.

Ex. What is the difference between while and do-while loop?

Solution:
While loop evaluates a test expression before allowing entry into the loop, whereas do-while loop is executed at least once before it evaluates the test expression which is available at the end of the loop.

While loop construct contains the condition first. If the condition is satisfied, the control executes the statements following the while loop else, it ignores these statements. The general form of while loop is

```
While(condition)
{
        statement1;
        statement2;
        ……
}
```

do-while loop construct is another method used in C programming. do-while loop ensures that the program is executed atleast once and checks whether the condition at the end of the do-while loop is true or false. As long as the test condition is true, the statements will be repeated. The control will come out from the loop, only when the test condition is false.

The do-while loop has the following form:

```
do
{
        statement1;
        statement2;
        ………..
}
while(condition);
```

The blocks of statements with in double braces { } following the word do are executed at least once. Then the condition is evaluated. If the condition is true, the block of statements are executed again until the value of condition tested is true.

Ex. What is the minimum number of iteration that

a. while loop could make ?
b. do-while loop could make ?

Solution:

a. while loop could make 0 iteration.
b. Do-while loop could make 1 iteration.

Q.6: Write a C program to generate the Fibonacci sequence
0, 1, 1, 2, 3, 5, 8,….. upto 100

Solution: The program to generate the Fibonacci sequence is given below:

```
/* To print Fibonacci series upto 100 */
#include<stdio.h>
#include<conio.h>
main( )
{
        int a = 0;
        int e = 1;
        int b = 1;
        int n, c = 1;
        clrscr();
        printf("The Fibonacci series upto 100 is:\n");
        printf("%d",a);
        do
        {
                printf("%5d",b);
                b = e+a;
                a = e;
                e = b;
                c = c+1;
        }
        while(b<=100);
        printf("\n\nPress any key to exit...\n");
        getch();
        return;
}
```

The output of the above program will be:

```
The Fibonacci series upto 100 is:
0   1   1   2   3   5   8  13  21  34  55  89

Press any key to exit...
```

Ex: Write a program to read in value of n and then print the first n Fibonacci numbers.

OR

Write a C program to print the nth fibonacci number.

Solution:

```
/* To print Fibonacci series */
#include<stdio.h>
main( )
{
    long int a = 0;
    long int e = 1;
    long int b = 1;
    int n, c = 1;
    printf("How many numbers you want in the Fibonacci series ? ");
    scanf("%d",&n);
    printf("%d\n",n);
    do
    {
            printf("%d\n",b);
            b = e+a;
            a = e;
            e = b;
            c = c+1;
    }
    while (n >= c);
}
```

Q. Give differences between **while** and **do-while** statement.

Solution: The differences between while and do-while statement are shown below:

while statement	**do-while statement**
1. The statement block in **while** statement is executed when the values of the condition is **true**.	1. The statement block is executed at least once irrespective of the value of the condition.
2. The condition is evaluated at the beginning of the loop. The statement block is executed if the value of the condition is **true**.	2. The condition is evaluated at the end of the loop. The statement block is executed again if the value of the condition is **true**.

Q. List a few unconditional control statement in C.
Solution: Statements, which are used to transfer the control to any part of the program without checking a condition, are referred as unconditional statement in C.

Unconditional control statement in C are:

(i) **break** statement
(ii) **continue** statement
(iii) **goto** statement
(iv) **exit( )** function.

goto Statement:

The **goto** statement is an unconditional transfer of control statement. It is used to transfer the control from one part of the program to another. The place to which the control is transferred is identified by a statement **label**. It has the following form.

goto label;

Where **label** is the statement label which is available anywhere in the program.

Eg.

```
----------------;
goto display;
----------------;
----------------;
display;
----------------;
```

When this statement is executed, the control is transferred to the statement label display which is followed by a comma.

break statement:

The **break** statement is used to transfer the control to the end of a statement block in a loop. It is an unavoidable statement to transfer the control to the end of a **switch** statement after executing any one statement block. It can be used within a for, while, do-while, or switch statement. It has the following form.

break;

Note that a program can be written without this statement other than in a **switch** statement.

Eg.

Consider the following example.

```
for(i = 1; i<= n; i++)
{
        -------------
        -------------
        break; ──────────┐
        ------------     │
        ------------     │
}                        │
---------------------◄───┘   control transferred to the end of the statement block.
```

continue Statement:

The **continue** statement is used to transfer the control to thebegining of a statement block in a loop.It has the following form.

continue;

Eg.

```
for(i=1; i<= 20;  i++)
    { ------------------
    Ch= getche();
    if (ch= ='C')
        {
        printf("\n C for continue is pressed");
        continue;
        }
         ---------------------
    }
```

Control is transferred to the beginning of the block

When this statement is executed, the control is transferred to the beginning of the loop such that the loop is repeated 20 times irrespective of the key pressed.

Exit () Function:

The exit () function is used to transfer the control to the end of a program(i.e. to terminate the program execution). It uses one argument in () and the value is zero for normal termination or non zero for abnormal termination.

Eg.

```
if( n < 0 )
{
        printf("\n  Factorial is not available for negative numbers");
        exit(0);
}
-----------------;
```

Note that the program execution is terminated when the value of the variable n is negative.

Return statement:

The keyword **return** is used to return any value from the function called. This statement terminates the function and returns a value to its caller. The **return** statements may also be used to exit from any function without returning any value. The **return** statement may or may not include an expression. The general format of using **return** statement is given below:

```
return;                 /* without expression */
return(expression)      /* with expression */
```

The above expression can be an integer value, a float or a char, this depends on the data type of the function declared.

Ex: Explain the functions of the following statements in C.

i) break
ii) return
iii) continue

OR

In what situations will you use a ‘break’ statement and a ‘continue’ statement? Explain with an example.

Solution: The uses of ‘return’, ‘break’ and ‘continue’ statements are given below:

Return statement: The keyword **return** is used to return any value from the function called. This statement terminates the function and returns a value to its caller. The **return** statements may also be used to exit from any function without returning any value. The **return** statement may or may not include an expression. The general format of using **return** statement is given below:

```
return;             /* without expression */
return(expression)  /* with expression */
```

The above expression can be an integer value, a float or a char, this depends on the data type of the function declared.

break statement: The break statement causes an immediate exit from the innermost loop structure or to exit from a switch. It can be used within a for, while, do-while, or switch statement.

- ➢ If we need to come out of a running program immediately without letting it to perform any further operation in a loop, then we can use break control statement.

The general format of the break statement is:

break; <Enter>

Following program segment is written to explain the use of break with while loop structure:

```
#include<stdio.h>
main( )
{
    int i, value;
    i=0;
    while(i<=10)
    {
            printf("Enter a number\n");
            scanf("%d",&value);
            if value = =0 || value<0)
            {
                    printf("Zero or negative value found.\n");
                    break;
            }
            i++;
```

```
        }
    }
```

The above program segment processes only the positive integers. Whenever the zero or negative value is found the program will display the message "Zero or negative value found" as an error and it will not execute the loop further.

continue statement: In some programming situations we want to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed. The keyword **continue** allows us to do this. When the keyword **continue** is encountered inside any C loop control automatically passes to the beginning of the loop.

A **continue** is usually associated with an **if**. As an example, let's consider the following program.

```
main( )
{
    int i, j;
    for(i=1;i<=2;i++)
    {
            for(j=1;j<=2;j++)
            {
                    if(i= =j)
                            continue;
                    printf("\n%d %d\n",i,j);
            }
    }
}
```

The output of the above program would be:

```
    1      2
    2      1
```

Note that when the value of **i** equals that of **j**, the **continue** statement takes the control to the for loop(inner) bypassing rest of the statements pending execution in the **for** loop(inner).

Ex: What are the uses of the following in C?
(i) break (ii) continue

Solution:

(i) **break:** A **break** statement is used to terminate the execution of a statement block. The next statement which follows this statement block will be executed. The keyword is **break**. When a **break** statement is executed in a loop, the repetition of the loop will be terminated.

(ii) **continue:** A **continue** statement is used to transfer the control to the beginning of a statement block. The keyword is **continue**. When a **continue** statement is executed in a loop, the execution is transferred to the beginning of the loop.

Ex: What is the similarity between break and continue statements?

Solution: Both break and continue are jump statements.

Ex: What is the function of break statement in a while, do-while or for loop?

Solution: If a break statement is included in a while, do-while or for loop, then control will immediately be transferred out of the loop when the break statement is encountered. This provides a convenient way to terminate the loop if an error or other irregular condition is detected.

Ex: Why the use of the goto statement should generally be avoided in a ‘C’ program?

Solution: The structured feature in ‘C’ requires that the entire program be written in an orderly, and sequential manner. For this reason, use of the goto statement should generally be avoided in a ‘C’ program.

Ex: Differentiate between break and exit( ).

Solution: break just terminates the execution of loop or switch in which it is written, whereas as exit( ) terminates the execution of the program itself.

Q. Distinguish between the **break** and **continue** statements in C.

Solution: The differences between the **break** and **continue** statements in C are listed below:

break statement	**continue statement**
1. A **break** statement is used to terminate the execution of a statement block. The next statement which follows this statement block will be executed. The keyword is **break**.	1. A **continue** statement is used to transfer the control to the beginning of a statement block. The keyword is **continue**.
2. When a **break** statement is executed in a loop, the repetition of the loop will be terminated.	2. When a **continue** statement is executed in a loop, the execution is transferred to the beginning of the loop.

Q. What are the differences between **break** and **exit( )** function.

Solution: The differences between break and exit( ) function are shown below:

break statement	**exit( ) function**
1. A **break** statement is used to terminate the execution of a statement block. The next statement which follows this statement block will be executed.	1. An **exit( )** function is used to terminate the execution of a C program permanently.
2. It is used in a program as **break**;	It is a built-in function and is used/called with necessary argument as **exit(0);**

Q. Write a program to calculate the sum of digits of a given 5 digit number.

Solution:

```
/* Sum of digits of a 5 digit number  */
#include<stdio.h>
#include<conio.h>
main()
{
        int num, a, n;
        int sum=0; /*sum initialised  to zero as otherwise it will contain a
                        garbage value */
        clrscr();
        printf("\nEnter a 5-digit number ");
        scanf("%d",&num);
        a=num%10; /*last digit extracted as remainder */
        n=num/10; /*remaining digits*/
        sum=sum+a; /*sum updated with addition of extracted digit */
        while(n!=0)
        {
        a=n%10; /* 4th digit */
        n=n/10;
        sum=sum+a;
        }
        printf("\nThe sum of the 5 digits of %d is %d",num, sum);
        printf("\n\n\nPress any key to exit...");
        getch();
        return;
}
```

Q. Write a program to print first n prime numbers using while loop.

Solution:

```
/* Generate first n prime numbers */
#include<stdio.h>
#include<math.h>
main()
{
      int n,num,d,t,count;
      printf("How many primes are required? ");
      scanf("%d",&n);
      printf("%d\n",n);
      printf("First %d primes are:\n");
      printf("%5d",2); /* 2 is the first and only even prime number */
      count=1;
      num=3; /* now start from 3 and test only odd numbers */
      while(count<n)
      {
            t=sqrt(num);
            d=2;
            while(d<=t)
            {
                  if(num%d==0)
                        break;
                   d++;
            }
            if(d>t)
```

```
                {
                        printf("%5d",num);
                        count++;
                }
                num+=2;
        }
}
```

RUN:

```
How many primes are required? 50
First 50 primes are:
    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  101  103  107  109  113  127  131
  137  139  149  151  157  163  167  173  179  181  191  193  197  199  211  223
  227  229
```

Q. Give an alternative for multiple **if** statement in C.

Solution: A **switch** statement can be used as an alternative for multiple **if** or nested **if** statements.

Consider the following example:

```
if(n = = 1)
          y = 1+x;
else
          if(n = = 2)
                    y = 1+x/n;
          else
                    if( n = = 3)
                              y = 1 +pow(x,n);
                    else
                              y = 1 + n * x;
```

The above program can also be written using a **switch** statement.

```
switch(n)
{
          case 1:        y = 1 + x;
                    break;
          case 2: y = 1 + x/n;
                    break;
```

```
        case 3: y = 1 + pow(x,n);
                break;
        default: y = 1 + n * x;
                break;
}
```

Q. Write a C program, which accepts a number and prints the sum of digits of this number.

Solution:

```
/* Accept a number and sum up the digit */
/* for example, 173456=1+7+3+4+5+6=26 */
#include<stdio.h>
main( )
{
        int n, rem, quo, sum=0;
        printf(“Enter a integer number: \n”);
        scanf(“%d”,&n);
        while(n>0)
        {
                rem=n%10;
                quo=n/10;
                sum=sum+rem;
                n=quo;
        }
        printf(“The sum of digit of digits which you entered is =
        %d”,sum);
}
```

Q. What is the use of **break** statement?

Solution: A **break** statement is used to transfer the control to the end of a statement block. The next statement, which follows this statement block, will be executed. Consider the following example.

```
for(i = 1; i<= n; i++)
```

```
{
        -------------
        -------------
        break;
        ------------
        ------------
}
 ---------------------        control transferred to the end of the statement block.
```

&&&&&&&&&