Covered Topics Under UNIT-3 of "PPS-PROGRAMMING FOR PROBLEM SOLVING (BCS101 / BCS201)"

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

Shweta Tiwari

Class Notes

Published Date: November, 2022

PPS: UNIT-3

Iteration and Looping

Array

FALL SEMESTER, YEAR (I/II sem, 1st yr)

FALL SESSION (2022-23)
(PPS)
MS. SHWETA TIWARI
Published Date: November, 2022

shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

—

By SHWETA TIWARI
Under On: UNIT-3

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Units Classified into Chapters

PPS: UNIT-3:

Chapter-6 LOOPING AND BRANCHING STATEMENTS

Chapter 6

LOOPING AND BRANCHING STATEMENTS

Loops are used to execute statement or block of statements more than once. In other words, Loop is defined as continuity. In simple language, it means to do a particular task again and again till a particular condition is true or satisfied.

For example

A person is given a task to go upstairs or downstairs for five no. of times continuously. These processes of going up and down again and again are known as loop. In a simple way we can say that a particular work or job is done repeatedly.

Use of loops in programming is to calculate the result or percentage of 100,000 students of Punjab school education board. In this case, we have to calculate the percentage of each student individually. To do this, we have to apply the same formula again and again for each student. This process is very time consuming, so to save memory space as well as time, loop is used in the database. We can apply formula with the help of loop in one single line according given condition.

Thus a loop is a part of program through which the control moves several times during the execution of the program. The part which is executed again and again is known as body of the loop. During the execution of loop, whenever control falls to end of the body of the loop, it again jumps to the starting point of body of the loop. This continues till a condition is satisfied and control comes out of the loop if the condition is not satisfied.

In 'C' we have three types of loops. These are known as:

1. **While**
2. **Do while**
3. **For**

In mostly every loop we have to identify 3 main parts of the loop.

a) initialization: Starting point of the loop

b) Condition:- Ending point of the loop

c) Increment/Decrement:- How will we reach from starting to end point of the loop.

1. While :- It executes the block of statements till the value of the conditional expression is true. In while if condition is true, then the body of the loop is executed. After execution of the body [contain statements and increment/decrement part], the test condition is again tested and if it is true, the body is executed again. This process is repeated till the test is true.

Syntax: -

```
While (expression)
{
Statement a;
Statement b;
Increment/decrement;
}
```

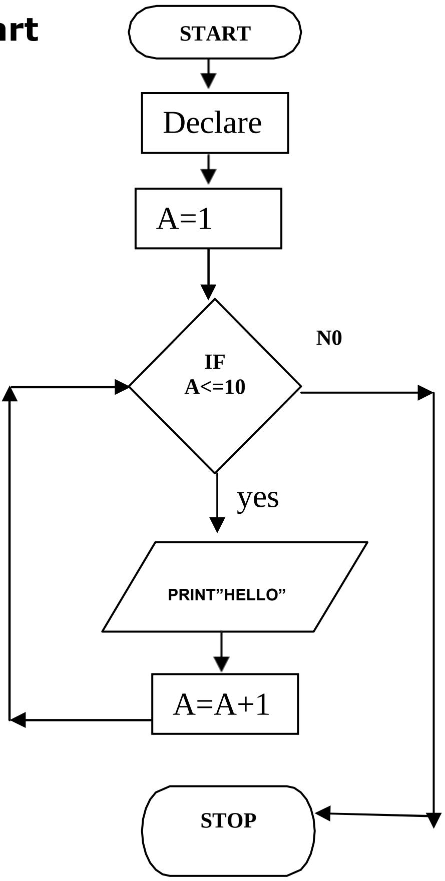
There will be no semicolon at the end of the line containing the keyword while in while loop.

1) Write an algorithm and draw flowchart to print “HELLO” 10 times.

```
Algorithm
        Step 1: START
        Step 2: DECLARE A
        Step 3: A=1
        Step 4: IF A<=10 GOTO STEP 5
 ELSE GOTO STEP 7
Step 5: PRINT “HELLO”
Step 6: A=A+1 GOTO STEP 4
        Step 7: STOP
```

Or

```
Algorithm
        Step 1: START
        Step 2: DECLARE A
        Step 3: A=1
        Step 4: Repeat while A<=10
PRINT “HELLO”
 A=A+1
End loop
        Step 7: STOP
```



Write A Program To Print 1 To 10 Series.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=1;
clrscr();
while(a<=10)
{
printf(“\n%d”,a);
a++;
}
getch();
}
```

a=1 is the starting point of the loop and condition written in while is the end point a++ is the increment. This represents that loop starts from 1 ends at 10 and increment 1. So output will be 1 2 3 4 5 6 7 8 9 10, similarly in following program loop starts from 2 ends at 100 with 2 increment.

Write A Program To Print Series 2,4,8,10 ---- 100

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=2;
clrscr();
while(a<=100)
{
printf(“%d”,a);
a=a*2;
}
getch();
}
```

Loops are also very helpful if we want to multiply , add or count number from the continue list like 2 4 6 8 10 12 14 16 18 20 to add numbers we use s=s+a where s is used for sum and a is the variable of the loop at start value of s should be initilized with 0 as in the following example value of s in the beging of the loop is 0 after one iteration of the loop value of s=s+1 means s=0+1 this make s=1 and after second itration s=1+2 so s is 3 similar after third itration s=3+3 (6) after fourth s=6+4 (10) this is executed 10 times. In the end value of s=1+2+3+4+5+6+7+8+9+10=55. Similarly if we want to count numbers we use c=c+1 and for multiply m=m*a.

Write a program to find sum of 1 to 10 numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a=1,s=0;
clrscr();
while(a<=10)
{
s=s+a;
a++;
}
printf(“%d”,s);
getch();
}
```

2. DO-WHILE: - It checks the condition after executing the statements so it executes the statements at least once even if the expression is false. Condition is checked at the end of the do while loop. Mostly this loop is used for software testing (Means whether the condition is true or false, loop will be execute once)

Syntax

```
  do
  {
  Statement a;
  Statement b;
  Increment/decrement;
  }
while (expression);
```

There will be no semicolon at the end of the line containing the keyword do, But we need to place semicolon at the end of keyword while.

1) Write an algorithm to print "hello" n times.

Algorithm

```
  Step 1: START
   Step 2: DECLARE i,num
   Step 3: PRINT "Enter the ending point number"
Step 4: INPUT num
   Step 5: i=1
   Step 6: PRINT "hello"
   Step 7: i=i+1
  Step 8: if(i<=num) then GOTO STEP 6 else GOTO STEP 9
  Step 9: STOP
```

Write A Program To Print hello N times

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i =1 ,num;
clrscr();
printf("enter number");
scanf("%d",&num);
 do
{
printf("hello");
i++;
}
while(i<=num);
getch();
}
```

Difference between do-while loop and while loop

In both the cases the condition is checked. In case of while, condition is checked before the execution of the statements but in case of do-while, statements are executed at least once whether the condition is true or false i.e. condition is checked at the end of the do-while block.

```
a=11;
while(a<=10)
{
   printf("%d",a);
   a++;
}
```

```
Output: [BLANK SCREEN]
```

```
a=11;
do
{
   printf("%d",a);
   a++;
} while(a<=10);
```

```
Output: 11
```

3. FOR: - In for loop the initialization, condition and increment or decrement can be done in one line. It executes the statements till the condition is true. Mostly this loop is used in complex programming.

Syntax

```
for (initialization; condition; increment/decrement)
{
Statement;
Statement;
}
```

1) First initialize the variable.
2) After initialization, condition is checked. If condition is true then statement is executed.
3) After that increment or decrement the variable according to condition.

Write a program to find sum of 1 to 10 numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,s=0;
clrscr();
for(a=1;a<=10;a=a+1)
{
s=s+a;
}
printf("%d",s);
getch();
}
```

Write a program to print factors of a number

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,c=0;
clrscr();
printf("enter a no");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(n%i==0)
{
printf(" %d",i);
}
}
getch();
}
```

Write a program to count factors of a number

To count factors we use formula c=c+1 the value of c is incremented every time we find a value of i that divide the number n. In the end of the loop the value of C is equal to the total factors of n.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,c=0;
clrscr();
printf("enter a no");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(n%i==0)
{
c=c+1;
}
}
printf("count= %d",c);
getch();
}
```

Write a program to print number is prime or not.

**Prime numbers are only divided by 1 or itself like 7, 11, 13, 17 etc.*

This program is same as last one only we add a condition in the end that if no of factors are 2 then no is prime else not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,c=0;
clrscr();
printf("enter a no");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(n%i==0)
{
c=c+1;
}
}
if(c==2)
printf("number is prime");
else
printf("number is not prime");
getch();
}
```

***Write a program to print sum of digits of a number*.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,s=0;
clrscr();
printf("enter a no");
scanf("%d",&n);   //let n=1234
while(n>0)
{
i=n%10;       //4         3     2     1
s=s+i;  //0+4=4     4+3=7 7+2=9 9+1=10
n=n/10; //123 12    1     0
}
printf("sum of digits =%d",s);      //so s=10
getch();
}
```

Write a program to print reverse of a number.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,s=0;
clrscr();
printf("enter a no");
scanf("%d",&n);   // let n=1234
while(n>0)
{
i=n%10; //i=4                 3              2              1
s=s*10+i;//s=0*10+4=44*10+3=4343*10+2=432432*10+1=432
n=n/10;
}
printf("reverse of a number =%d",s);
getch();
}
```

Write a program to print no is Armstrong or not.

*153 is an Armstrong number because
1*1*1 =   1
5*5*5 = 125
3*3*3 =   27
Total   157

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,s=0,m;
clrscr();
printf("enter a no");
scanf("%d",&n);
m=n;
while(n>0)
{
i=n%10;
s=s+i*i*i;
n=n/10;
}
if(s==m)
printf("number is Armstrong”);
else
printf("number is not Armstrong”);
getch();
}
```

1. **NESTED LOOP:** -Nested loop means nesting or loops means within loop. In nested loop the first loop is called outer loop & second loop is called inner loop. In this the condition of outer loopis true then the condition of inner loop is checked. If it is true then inner loop will execute. The inner loop would act as a part of the body for the outer loop. Therefore, execution of inner loop will depend on the execution of the outer loop.

Syntax

```
    for (initialization; condition; increment/decrement) //outer
                                                  Loop
{
                for (initialization; condition; increment/decrement)
                                               // Inner
           loop
            {
                Statement;
            }
    }
```

Why we use nested loops:- If we want to find sum of 5 subjects of a student we use single loop and if we want to find sum of 5 students in 5 subjects each then we need to execute the loop 5 times so that we can find sum of 5 students for this we need nested loops. E.g

```
    For(stu=1;stu<=5;stu=stu+1)
    {
        for(sub=1;sub=1;sub=sub+1)
        {
            statements;
        }
    }
```

If the condition of outer loop is true then control will goto the inner loop and if the condition of inner loop is true then executes the statement. If the condition of outer loop is false the control doesn't go into in the inner loop.

Write a program to print :-

```
                12345
          12345
          12345
          12345
          12345
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=5;i++)
{
                for(j=1;j<=5;j++)
                         {
                                  printf(“%d”,j);
                         }
                printf(“\n”);
        }
getch();   }
```

In above program we have used nested loop. Ist loop is outer loop and 2nd loop is inner loop. If the outer loop is true then control goes to the inner loop otherwise exit the control in outer loop. In simple meaning outer loop act as no of rows and inner as no of column.

Write a program to print 1

```
12
123
1234
12345
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=5;i++)
{
             for(j=1;j<=i;j++)
                      {
                               printf(“%d”,j);
                      }
             printf(“\n”);
    }
getch();
}
```

Write a program to print 1

```
22
333
4444
55555
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
```

```
clrscr();
```

```
for(i=1;i<=5;i++)
{
            for(j=1;j<=i;j++)
                     {
                              printf("%d",i);
                     }
            printf("\n");
   }
getch();
}
```

Write a program to print 9
99
999
9999

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j;
clrscr();
for(i=1;i<=5;i++)
{

            for(j=1;j<=i;j++)
                     {
                              printf("9");
                     }
            printf("\n");
   }
getch();
}
```

Write a program to print 1 to 1000 palindrome numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,s=0,m;
clrscr();
for(m=1;m<=1000;m++)
{
n=m;
s=0;
while(n>0)
{
i=n%10;
s=s*10+i;
n=n/10;
}
if(s==m)
printf(" %d",s);
}
getch();
}
```

Branching Statements

There are another kind of control statements that can change the execution of loop or execution of the program. Branching statements are further divided into three types.

a. **Break**
b. **Continue**
c. **goto**

Break: - Break statement is used to exit from the loop or switch statement. Whenever break statement is encountered the rest of the statements inside the loop are ignored and the control go to the next statements after the loop. It can also be used with a while , a do while , a for or a switch statement.

EXAMPLE BREAK STAEMENT

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=1;i<=10;i++)
{
                if(i==5)
    {
                          break;
    }
printf("\n% d",i);
}
getch();
}
```

output

```
1
2
3
4
```

***exit** function:- It terminates the whole program instead of loop..*

Continue: -The continue statement is used to skip the remaining statements of the body of the loop where it defined, it does not terminate the loop but the control is transferred to the start of the next loop iteration. Continue statement is used to only break the current iteration. After continue statement the control returns to the top of the loop test conditions.

Example continue statement

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrcsr();
for(i=1;i<=10;i++)
{
    if(i==5)
      {
              continue;
      }
printf(" %d ",i);
}
getch();
}
```

Output: 1 2 3 4 6 7 8 9 10

Goto: - The goto statement is control statement that causes the control to jump to a different location in the program with or without checking any condition . It is always used with the label. It increase the readable complexity of the program so programmers avoid to use this. The *goto* statement can only be used within the body of a function definition.

There are two types of goto:-

1) Unconditional jump
2) Conditional jump

1. **Unconditional Jump**: In this control is transferred from one statement to another without any condition.

EXAMPLE Unconditional GOTO STATEMENT

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
printf(“\none”);
printf(“\nTwo”);
printf(“\nThree”);
goto abc;
printf(“\Four”);
printf(“\Five”);
printf(“\nSix”);
abc:
printf(“\nSevan”);
printf(“\nEight”);
getch();
}
```

Output

```
One
Two
Three
Seven
Eight
```

2. **Conditional Jump**: In this control is transferred from one statement to another according to the condition.

EXAMPLE conditional GOTO STATEMENT

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a;
Abc:
printf(“Enter no greater than 10”);
scanf(“%d”,&a);
if(a<=10)
goto Abc;
printf(“You enter number greater than 10”);
getch();
}
```

in this program control goes back to Abc label until user not enter number greater than 10.

Important Questions[Short]

1. What do you mean by loop?
2. What are the limitations of goto statement?
3. Write advantages of for loop over while and do-while.

Important Questions[Long]

1. Explain various types of loops in 'C' language .
2. Write difference between break and continue.
3. Write difference between while and do-while.
4. What are various jump statements in 'C' language .

***** Perform Programs from 27 to 51 from program list***