CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

INSTITUTE - UIE

DEPARTMENT- ACADEMIC UNIT-2

Bachelor of Engineering (Computer Science & Engineering)

Subject Name: Introduction to Problem Solving

Code:22CSH-101

Loop Control structure in C

Introduction to Problem Solving

Course Objectives

The course aims to provide exposure to problem-solving through programming.
The course aims to raise the programming skills of students via logic building capabil ity.
With knowledge of C programming language, students would be able to model real world problems.

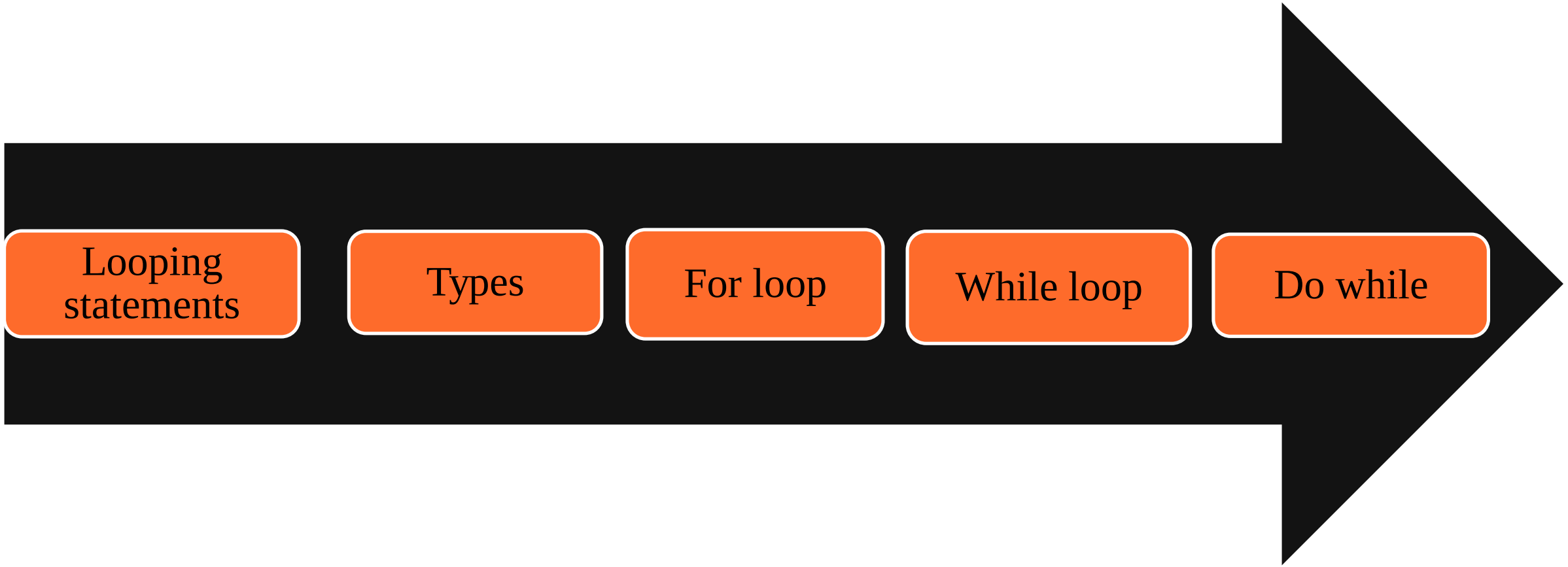
Learning Outcomes

CO Number	Course Outcome
CO1	Remember the concepts related to fundamentals of C language, draw flowcharts and write algorithm/pseudocode.
CO2	Understand the way of execution and debug programs in C language.
CO3	Apply various constructs, loops, functions to solve mathematical and scientific problem.
CO4	Analyze the dynamic behavior of memory by the use of pointers.
CO5	Design and develop modular programs for real world problems using control structure and selection structure.

Scheme of Evaluation

S.No	Type of Assessment	Weightage of actual conduct	Frequency of the task per semester	Final Weightage	Remarks
1	Practical Worksheet/ Practical Projects and Practical Learnings (Continuous Assessment) (Practical Component)	30 marks for each experiment	8-12 experiments	45	
2	Portfolio/ Discussion forum (Practical Component)	-------	Engagement Task : Non-Graded	-----	
3	Practical Mid-Term Test (Practical Component)	15	1 per semester	15	
4	Assignment/ Presentation/ Group Discussion etc. (Theory Component)	10 marks of each assignment	3 (One Per Unit)	10	
5	Time bound Surprise test (Theory Component)	12 marks for each test	3 (One Per Unit)	4	
6	Quiz (Theory Component)	4 marks of each quiz	6 (Two per Unit)	4	
7	Mid-Semester Test (Theory Component)	20 marks for each MST	2 per semester	20	
8	Attendance Score	---	---	2	Calculated based on collective Attendance of subject code.
9	Practical End Term Evaluation (Practical Component)	40	1 per semester	40	At the end of Semester
10	Theory End term exam (Theory Component)	60	1 per semester	60	At the end of Semester

Contents



Looping Statements

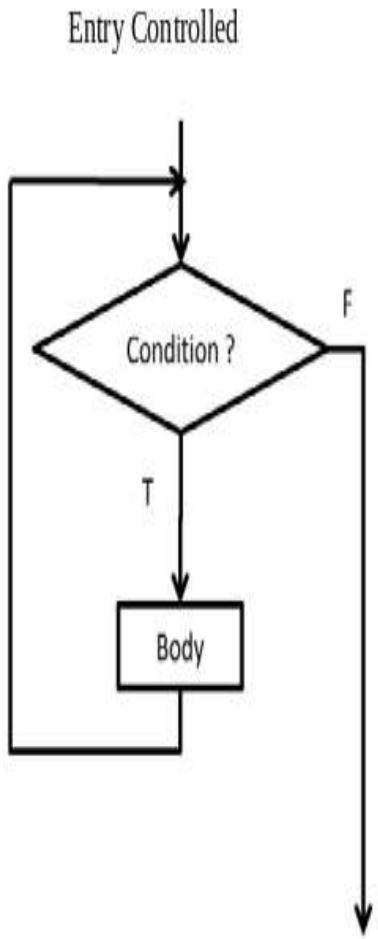
The statements that help us to execute set of statements repeatedly are called as looping statements. It executes a block of **statements** number of times until the condition becomes false.

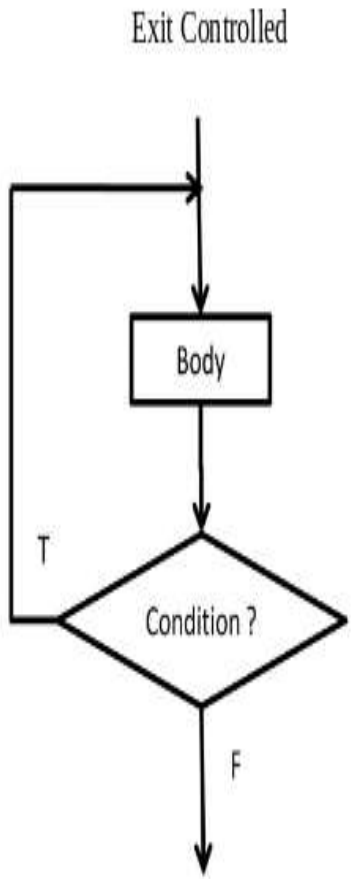
It executes a block of **statements** number of times until the condition becomes false.

Types of Looping statements

- **Entry controlled loop or Pre-Test Loop**: In the entry controlled loop or Pre-Test loop, the conditi on is checked before we start and at the beginnin g of each iteration of the loop. If the condition is t rue, we execute body of loop; if the control expre ssion is false, we terminate the loop.
- **Examples**: while statement and for statement

- **Exit controlled loop or Post-Test Loop:** In the exit controlled loop or Post-Test loop, the conditio n is checked after or at the end of each iteration of the loop. If the condition is true, we repeat bod y of loop; if the condition is false, we terminate th e loop.
- **Examples:** do…while statement





Loops Definition & Syntax

Statements	Definition & Syntax

while	It is a most basic loop in C programming. It has one control condition, and executes as long the condition is true. The condition of the loop is tested before the body of th e loop is executed.
	Syntax:
	while (condition)
	{
	statement(s);
	Incrementation;
	}

for	C for loops is very similar to a while loops in that it continues to process a block of co de until a statement becomes false, and everything is defined in a single line. The for loop is also entry
	-controlled loop.
	Syntax:
	for (init; condition; increment)
	{
	statement(s);
	}

Do while	C do while loops are very similar to the while loops, but it always executes the code block at least once and furthermore as long as the condition remains true.
	Syntax:
	do
	{
	statement(s);
	}while(condition);

Working:
step1: The loop variable is initialized with some value and then it has been tested for the condition.
step2: If the condition returns true then the statements inside the body of while loop are executed else control comes out of the loop.
step3: The value of loop variable is incremented/decremented then it has been tested again for the loop condition.

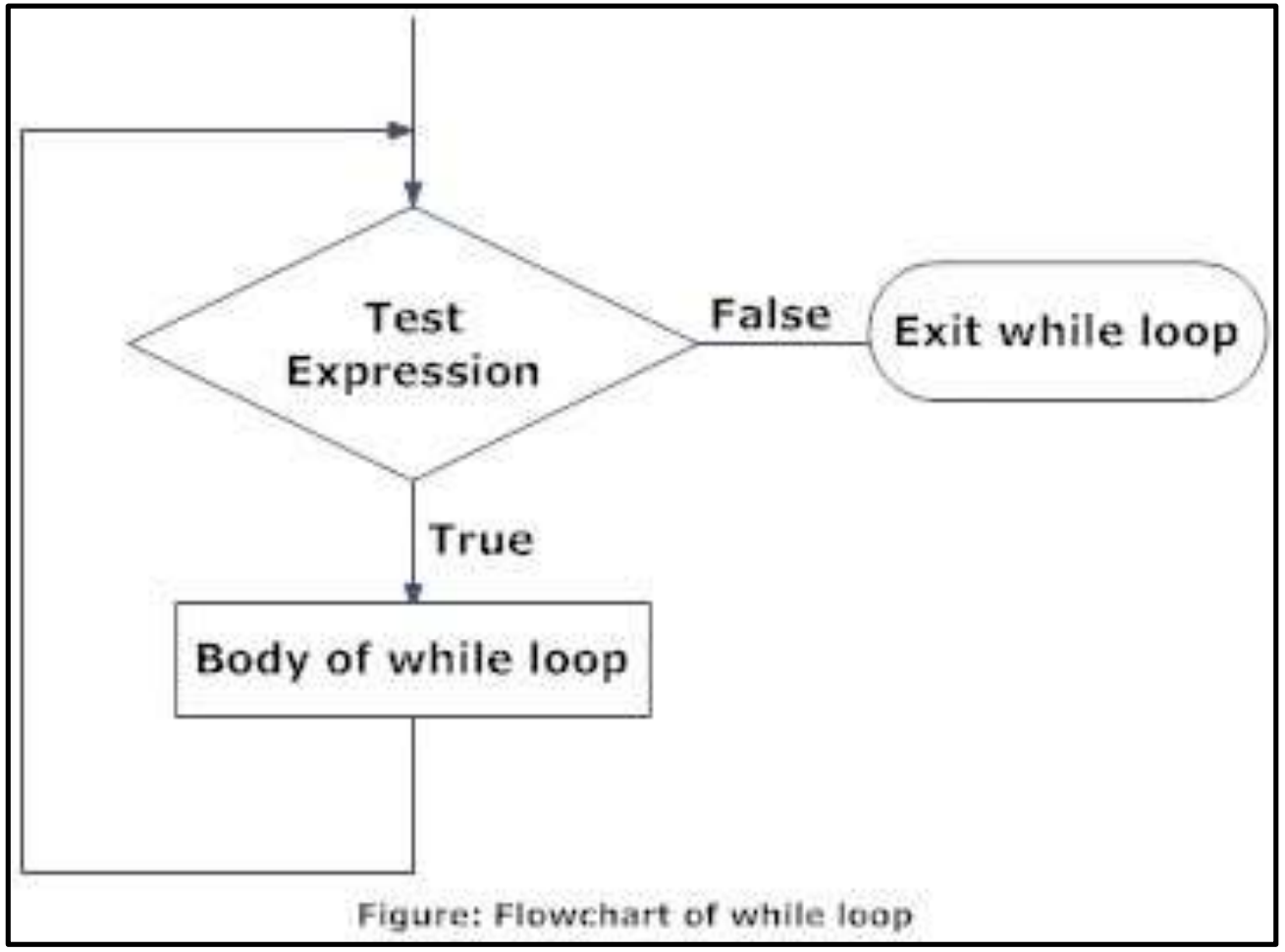


Figure: Flowchart of while loop

Example 1

Example 1 Program to print first 10 natural numbers.

```
#include<stdio.h>
void main( )
{
    int x;
    x = 1;
    while(x <= 10)
   {
        printf("%d\t", x);
        x++;
    }
}
```

```
gcc -o /tmp/qXrGQORDYe.o /tmp/qXrGQORDYe.c -lm
/tmp/qXrGQORDYe.o
1    2    3    4    5    6    7    89   10
```

1
1

Example 2

```c
#include<stdio.h>
int main()
{
    int i=1;
    while(i<=5)

        printf("hello\n");
        i++;
    return 0;
}
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

Infinite loop

Do-while loop

Working:

1. First we initialize our variables, next it will enter into the Do While loop.
2. It will execute the group of statements inside the loop.
3. Next we have to use Increment and Decrement Operator inside the loop to increment or decrements the value.
4. Now it will check for the condition. If the condition is True, then the statements inside the do while loop will be executed again. It will continue the process as long as the condition is True.
5. If the condition is False then it will exit from the loop.

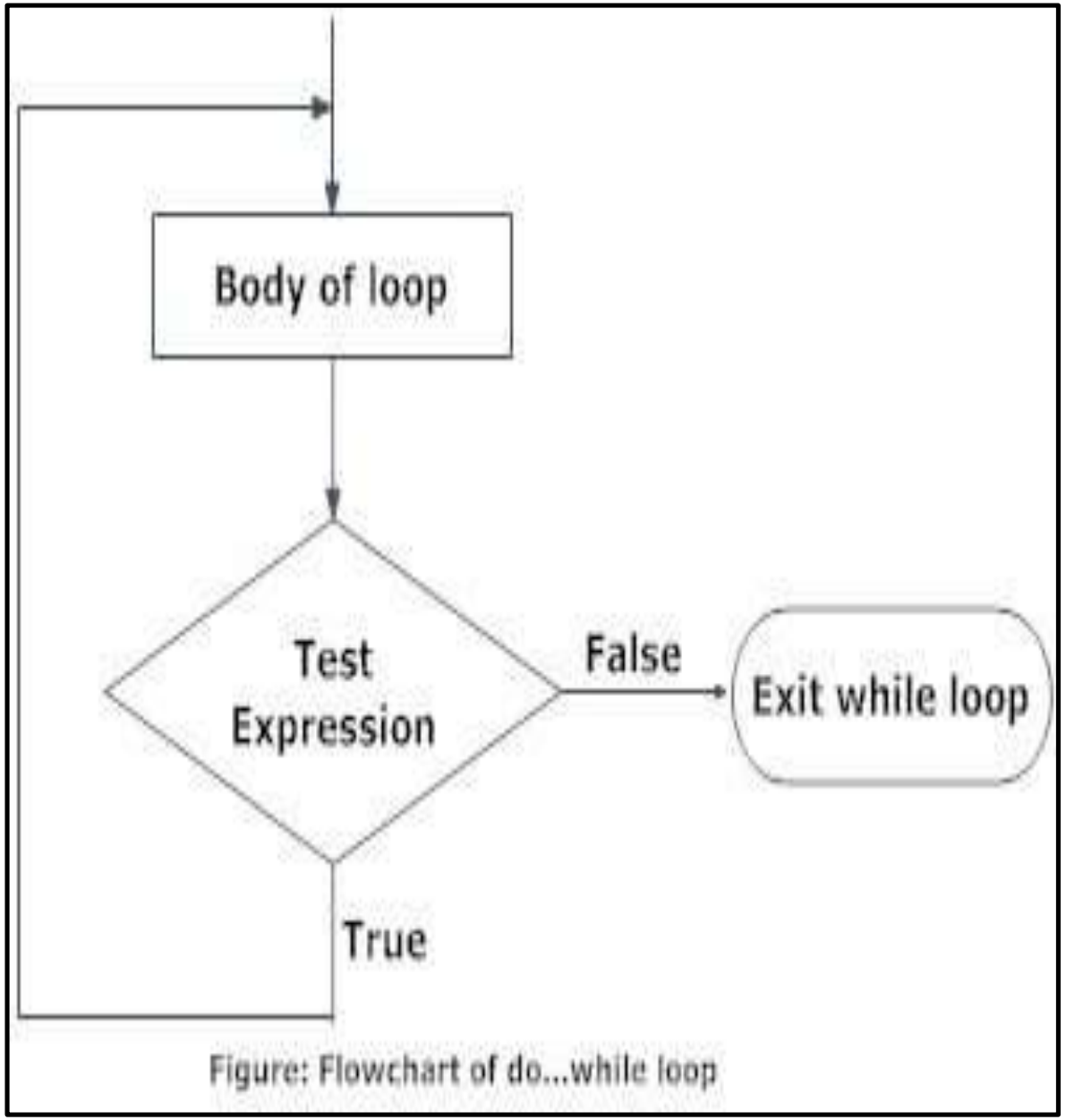


Figure: Flowchart of do...while loop

Example

Program to print fibonacci series

```
#include<stdio.h>
void main()
{
    int n,f,f1=-1,f2=1;
    printf("  Enter The Number Of Terms:");
    scanf("%d",&n);
    printf("  The Fibonacci Series is:");
    do
    {
        f=f1+f2;
        f1=f2;
        f2=f;
        printf("  \n %d",f);
        n--;
    }while(n>=0);
    }
```

For loop

Working:

Step 1: First initialization happens and the counter variable gets initialized.

Step 2: In the second step the conditio n is checked, where the counter variab le is tested for the given condition, if th e condition returns true then the C stat ements inside the body of for loop gets executed, if the condition returns false then the for loop gets terminated and t he control comes out of the loop.

Step 3: After successful execution of s tatements inside the body of loop, the counter variable is incremented or decr emented, depending on the operation (++ or –)

Flowchart of continue statement

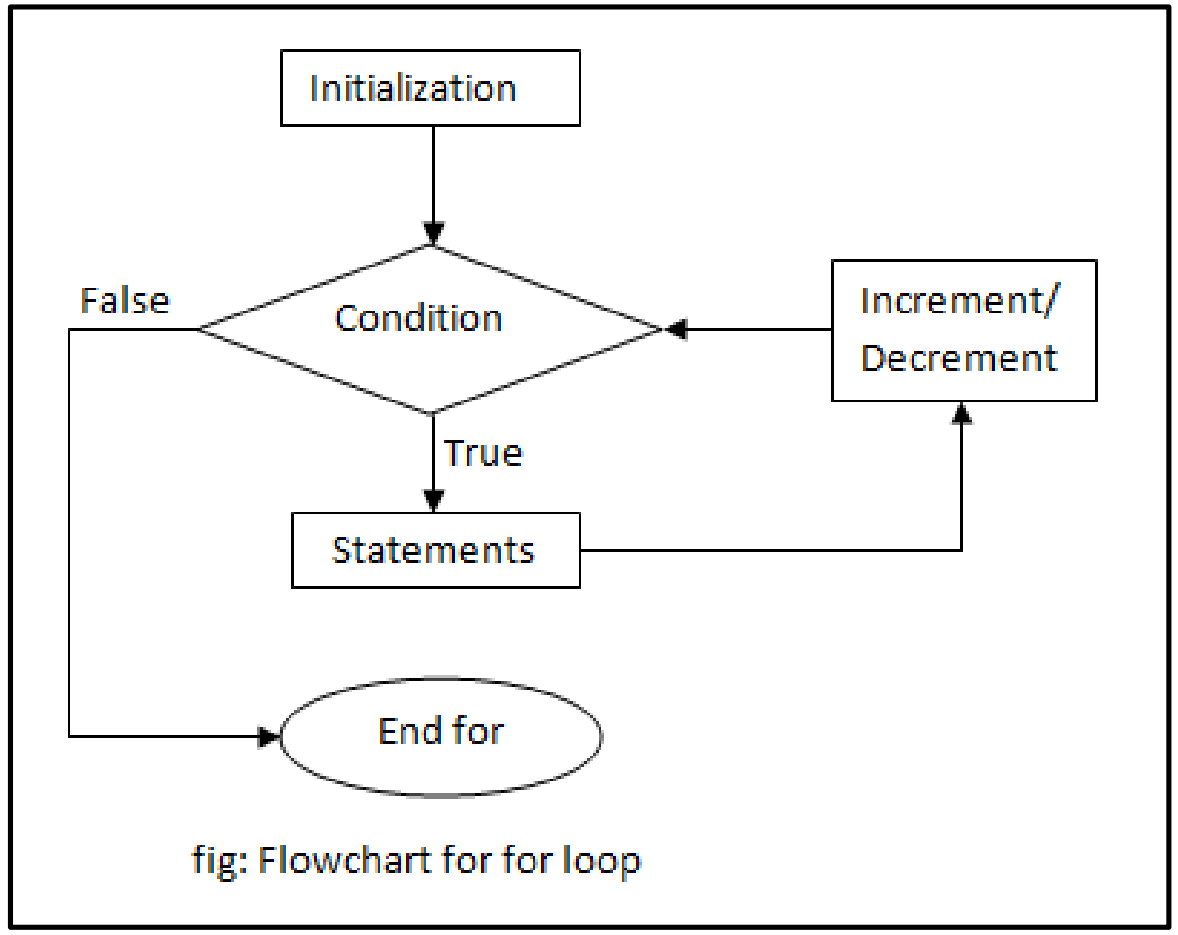


fig: Flowchart for for loop

Example

Program to print factorial of a number

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int fact, i, n;
   fact = 1;
   printf("Enter the number\t");
scanf("%d" , &n);
  for(i = 1; i <= n; i++)
   {
       fact = fact*i;
   }
 printf("Factorial of %d is %d", n , fact);
  getch();
}
```

Output

```
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
Enter the number 6
6
Factorial of 6 is 720
```

for	**It continues to process a block of code until a statement becomes false, and everything is defined in a single line.**
while	**It has one control condition, and executes as long the condition is true.**
Do while	**It always executes the code block at least once and furthermore as long as the condition remains true.**

FAQ

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

While	*Do-while*
1. Condition is at top.	1. Condition is at the bottom.
2. No necessity of bracket if there is single statement in body.	2. Brackets are compulsory even if there is a single statement.
3. There is no semicolon at the end of while.	3. The semicolon is compulsory at the end do-while.
4. Computer executes the body if and only if condition is true.	4. Computer executes the body at least once even if condition is false.
5. This should be used when condition is more important.	5. This should be used when the process is important.
6. This loop is also refered as entry controlled loop.	6. This loop is also refered as exit controlled loop.

7.While(n<10)
{
printf("%d\n",n);
}
	7.Do
	{
	Printf("%d\n",n);
	}while(n<=100);

- **Q2 Write a program to print table of a number enter by user.**

```
#include <stdio.h>
 void main()
{
   int j,n;
   printf("Input the number (Table to be calculated) : ");
   scanf("%d",&n);
   printf("\n");
   for(j=1;j<=10;j++)
   {
     printf("%d X %d = %d \n",n,j,n*j);
    }
      }
```

Output

```
gcc -o /tmp/JfA6xEVTs5.o /tmp/JfA6xEVTs5.c -lm
/tmp/JfA6xEVTs5.o
Input the number (Table to be calculated) : 9
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
9 X 10 = 90
```

Assessment Questions

1. Program to check Armstrong number.
2. Write a c program to check a number is prime number or not.
3. Write programs for the following :
   a) program to read 10 numbers from keyboard and find their sum and average.
   b) to calculate the simple interest.
4. do-while loop terminates when conditional expression returns?

A One
B ZERO
C NON-ZERO
D NONE OF THE ABOVE

5. What will be the output of following program ?

```
#include <stdio.h>
void main()
{
    int cnt=1;
    do
    {  printf("%d,",cnt);
        cnt+=1;
    }while(cnt>=10);
    printf("\nAfter loop cnt=%d",cnt);
    printf("\n")
```

Program takes an integer from the user and calculates the number of digits.
For example: If the user enters 231967, the output of the program will be 6.

References

Book References:
http://www2.cs.uregina.ca/~hilder/cs833/Other%20Reference%20Materials/The%20C%20Programming%20Language.pdf
http://www.freebookcentre.net/programming-books-download/The-Basics-of-C-Programming.html

Vedio Lecture: https://www.studytonight.com/c/loops-in-c.php
https://www.youtube.com/watch?v=4gFfGzpDGFw
https://spocathon.page/video/lecture-20-implementation-loops-statement-contd
https://study.com/academy/lesson/nesting-loops-statements-in-c-programming.html

Websites: https://www.programiz.com/c-programming/c-for-loop
https://www.tutorialspoint.com/cprogramming/c_loops.htm
https://beginnersbook.com/2014/01/c-loops-examples/

THANK YOU

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

INSTITUTE - UIE

DEPARTMENT- ACADEMIC UNIT-2

Bachelor of Engineering (Computer Science & Engineering)

Subject Name: Introduction to Problem Solving

Code:22CSH-101

Nested Loops

Introduction to Problem Solving

Course Objectives

The course aims to provide exposure to problem-solving through programming.
The course aims to raise the programming skills of students via logic building capability.
With knowledge of C programming language, students would be able to model real world problems.

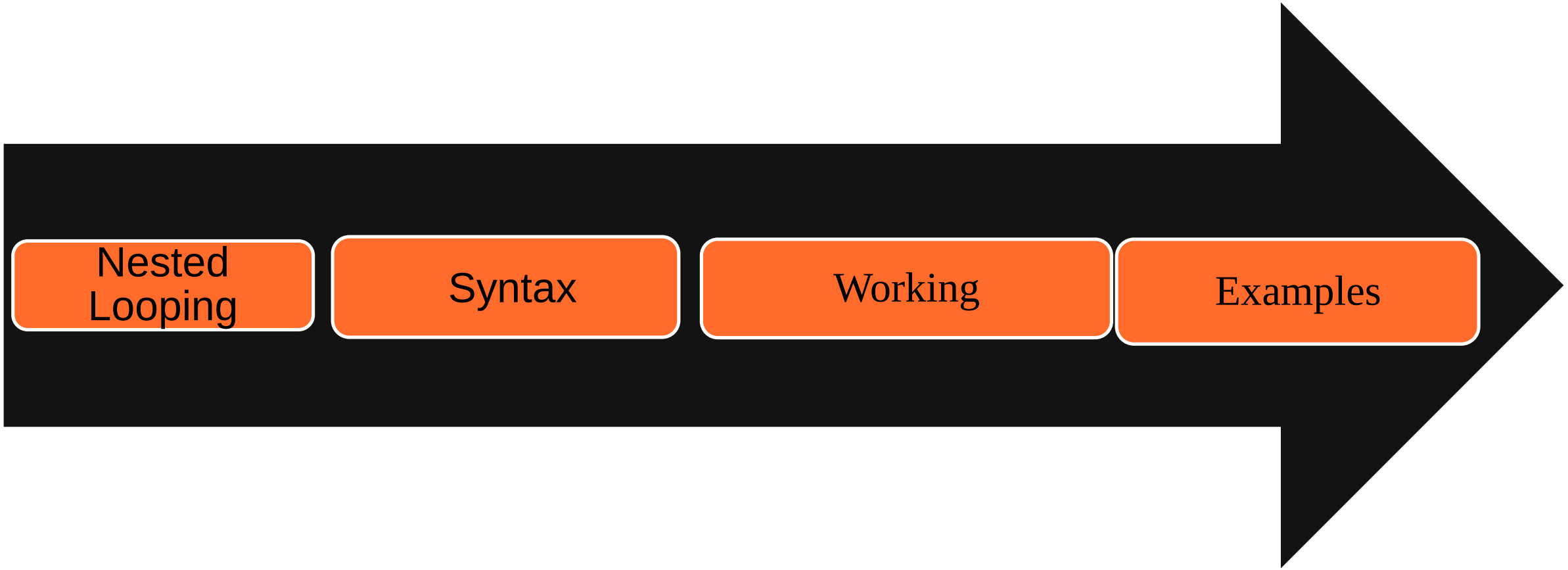
Learning Outcomes

CO Number	Course Outcome
CO1	Remember the concepts related to fundamentals of C language, draw flowcharts and write algorithm/pseudocode.
CO2	Understand the way of execution and debug programs in C language.
CO3	Apply various constructs, loops, functions to solve mathematical and scientific problem.
CO4	Analyze the dynamic behavior of memory by the use of pointers.
CO5	Design and develop modular programs for real world problems using control structure and selection structure.

Scheme of Evaluation

S.No	Type of Assessment	Weightage of actual conduct	Frequency of the task per semester	Final Weightage	Remarks
1	Practical Worksheet/ Practical Projects and Practical Learnings (Continuous Assessment) (Practical Component)	30 marks for each experiment	8-12 experiments	45	
2	Portfolio/ Discussion forum (Practical Component)	-------	Engagement Task : Non-Graded	-----	
3	Practical Mid-Term Test (Practical Component)	15	1 per semester	15	
4	Assignment/ Presentation/ Group Discussion etc. (Theory Component)	10 marks of each assignment	3 (One Per Unit)	10	
5	Time bound Surprise test (Theory Component)	12 marks for each test	3 (One Per Unit)	4	
6	Quiz (Theory Component)	4 marks of each quiz	6 (Two per Unit)	4	
7	Mid-Semester Test (Theory Component)	20 marks for each MST	2 per semester	20	
8	Attendance Score	---	---	2	Calculated based on collective Attendance of subject code.
9	Practical End Term Evaluation (Practical Component)	40	1 per semester	40	At the end of Semester
10	Theory End term exam (Theory Component)	60	1 per semester	60	At the end of Semester

Contents



Nested Looping

A loop within another loop is called nested loop. C programming language supports nesting of one loop inside another. We can define any number of loops inside another loop. And also have any number of nesting level..

Put any type of loop in another type. Write for loop inside while loop, while inside another while etc. A programmer nest up to 3 loops. But there is no limit of nesting in C.

Loops Definition & Syntax

Syntax

```
 outer_loop
{
 inner_loop
 {
   // Inner loop statement/s
 }
 // Outer loop statement/s
}
```


How Nested loop work:

There are two conditions that are given.

1. The inner loop condition gets executed only when the outer loop condition gives the Boolean output as True. Else the flow control directly goes out of both the loops.
2. Now coming into the execution of the inner loop, If the loop condition gives a true result, then the block of statements under that loop and the incremental condition gets executed.
3. And in turn, if the condition gives a Boolean condition as False, then the inner loop gives its control ba ck to the outer loop and again same conditions/loops gets executed/repeated.

Example1 of Nested do-while loop

```
#include <stdio.h>
int main()
{
  int i=1;
do          // outer loop
{
    int j=1;
    do      // inner loop
   {
      printf("*");
      j++;
   }while(j<=8);
    printf("\n");
    i++;
     }while(i<=4);
}
```

Explanation:

•First, we initialize the outer loop counter variable, i.e., 'i' by 1.

•As we know that the do..while loop executes once without checking the condition, so the inner loop is executed without checking the condition in the outer loop.

•After the execution of the inner loop, the control moves to the update of the i++.

•When the loop counter value is incremented, the condition is checked. If the condition in the outer loop is true, then the inner loop is executed.

This process will continue until the condition in the outer loop is true

Output:

```
Output
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
********
********
********
********
```

Example2 using Nested for loop

```c
#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        printf("\nFor i = %d ",i);
        for(j=0;j<i;j++)
        {
            printf("*");
        }
    }
    return(0);
}
```

Output

```
For i = 1 *
For i = 2 **
For i = 3 ***
For i = 4 ****
For i = 5 *****
--------------------
```

1
1

Nested Looping

A loop within another loop is called nested loop. C programming language supports nesting of one loop inside another.

We can define any number of loops inside another loop. And also have any number of nesting level.

Write for loop inside while loop, while inside another while etc. A programmer nest up to 3 loops. But there is no limit of nesting in C.

It always executes the code block at least once and furthermore as long as the condition remains true.

FAQ

Q1 Print the pattern

```
#include<stdio.h>
int main()
{
int i,j;
for(j=1;j<=5;j++)
{
  for(i=1;i<=j;i++)
     printf("%5d",i);
  printf("\n\n");
}
for(j=4;j>=1;j--)
{
   for(i=1;i<=j;i++)
      printf("%5d",i);
   printf("\n\n");
}
return 0;
}
```

1
3

FAQ

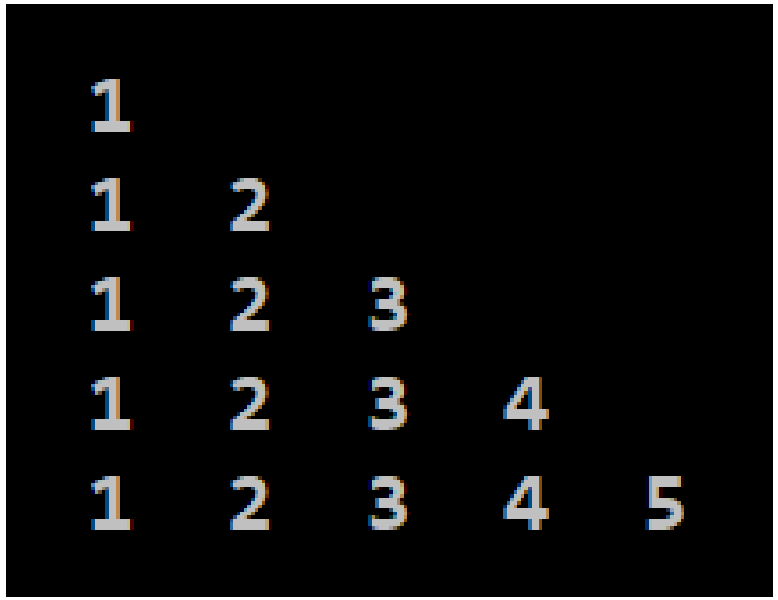
- **Q2 Print the reverse patter**

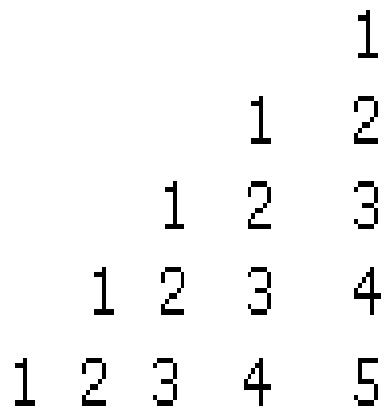
```
#include<stdio.h>
int main()
{
int i,j;
for(j=5;j>=1;j--)
{
   for(i=j;i>=1;i--)
     printf("%5d",i);
   printf("\n\n");
}
return 0;
}
```

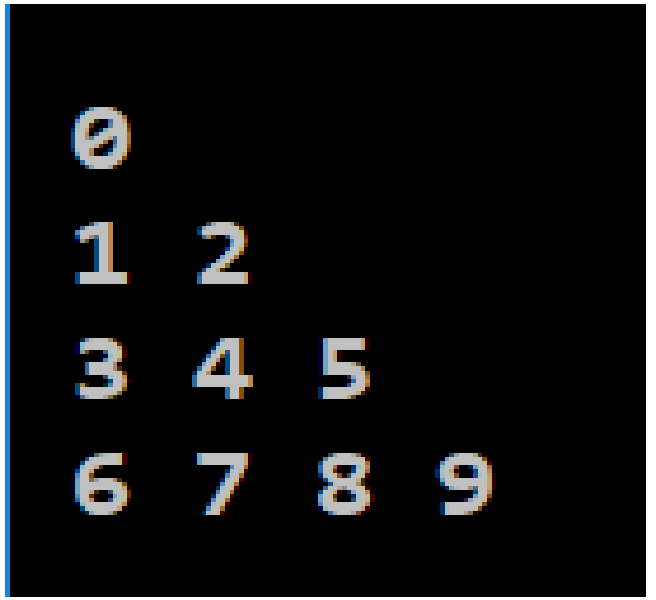
Output

```
gcc -o /tmp/MTiRIwHYT9.o /tmp/MTiRIwHYT9.c -lm
/tmp/MTiRIwHYT9.o
5    4    3    2    1

4    3    2    1

3    2    1

2    1

1
```

Assessment questions







Print the above two numbering patterns

References

Book References:
http://www2.cs.uregina.ca/~hilder/cs833/Other%20Reference%20Materials/The%20C%20Programming%20Language.pdf
http://www.freebookcentre.net/programming-books-download/The-Basics-of-C-Programming.html

Vedio Lecture: https://study.com/academy/lesson/nesting-loops-statements-in-c-programming.html
https://www.youtube.com/watch?v=mZqo8KDR37U
https://nptel.ac.in/courses/106/104/106104128/

Websites: https://www.studytonight.com/c/programs/loop/nested-loops
https://beginnersbook.com/2014/01/c-for-loop/
https://www.tutorialspoint.com/cprogramming/c_nested_loops.htm

THANK YOU

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

INSTITUTE - UIE

DEPARTMENT- ACADEMIC UNIT-2

Bachelor of Engineering (Computer Science & Engineering)

Subject Name: Introduction to Problem Solving

Code:22CSH-101

Jump Statements

Introduction to Problem Solving

Course Objectives

The course aims to provide exposure to problem-solving through programming.
The course aims to raise the programming skills of students via logic building capability.
With knowledge of C programming language, students would be able to model real world problems.

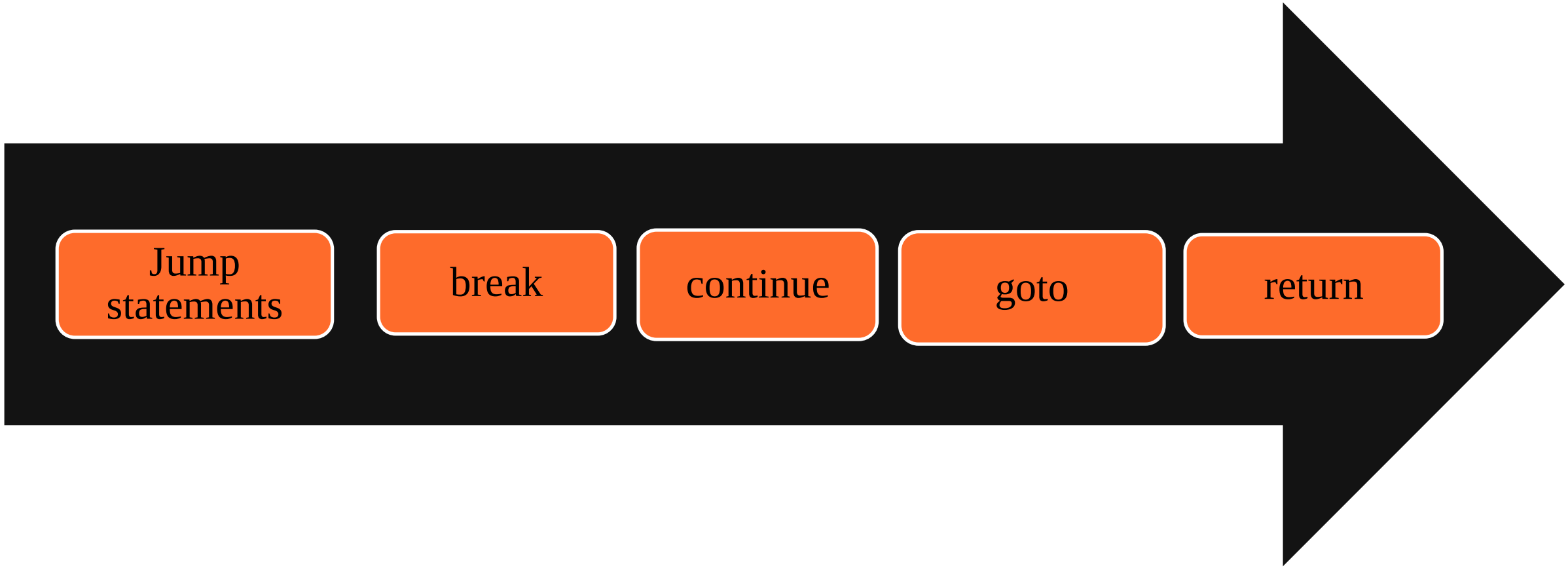
Learning Outcomes

CO Number	Course Outcome
CO1	Remember the concepts related to fundamentals of C language, draw flowcharts and write algorithm/pseudocode.
CO2	Understand the way of execution and debug programs in C language.
CO3	Apply various constructs, loops, functions to solve mathematical and scientific problem.
CO4	Analyze the dynamic behavior of memory by the use of pointers.
CO5	Design and develop modular programs for real world problems using control structure and selection structure.

Scheme of Evaluation

S.No	Type of Assessment	Weightage of actual conduct	Frequency of the task per semester	Final Weightage	Remarks
1	Practical Worksheet/ Practical Projects and Practical Learnings (Continuous Assessment) (Practical Component)	30 marks for each experiment	8-12 experiments	45	
2	Portfolio/ Discussion forum (Practical Component)	-------	Engagement Task : Non-Graded	-----	
3	Practical Mid-Term Test (Practical Component)	15	1 per semester	15	
4	Assignment/ Presentation/ Group Discussion etc. (Theory Component)	10 marks of each assignment	3 (One Per Unit)	10	
5	Time bound Surprise test (Theory Component)	12 marks for each test	3 (One Per Unit)	4	
6	Quiz (Theory Component)	4 marks of each quiz	6 (Two per Unit)	4	
7	Mid-Semester Test (Theory Component)	20 marks for each MST	2 per semester	20	
8	Attendance Score	---	---	2	Calculated based on collective Attendance of subject code.
9	Practical End Term Evaluation (Practical Component)	40	1 per semester	40	At the end of Semester
10	Theory End term exam (Theory Component)	60	1 per semester	60	At the end of Semester

Contents



Jump Statements

Jump statements can be used to modify the behavior of conditional and iterative statements

They allow you to exit a loop, start the next iteration of a loop, or explicitly transfer program control to a specified location in your program.

Types of Jump statements

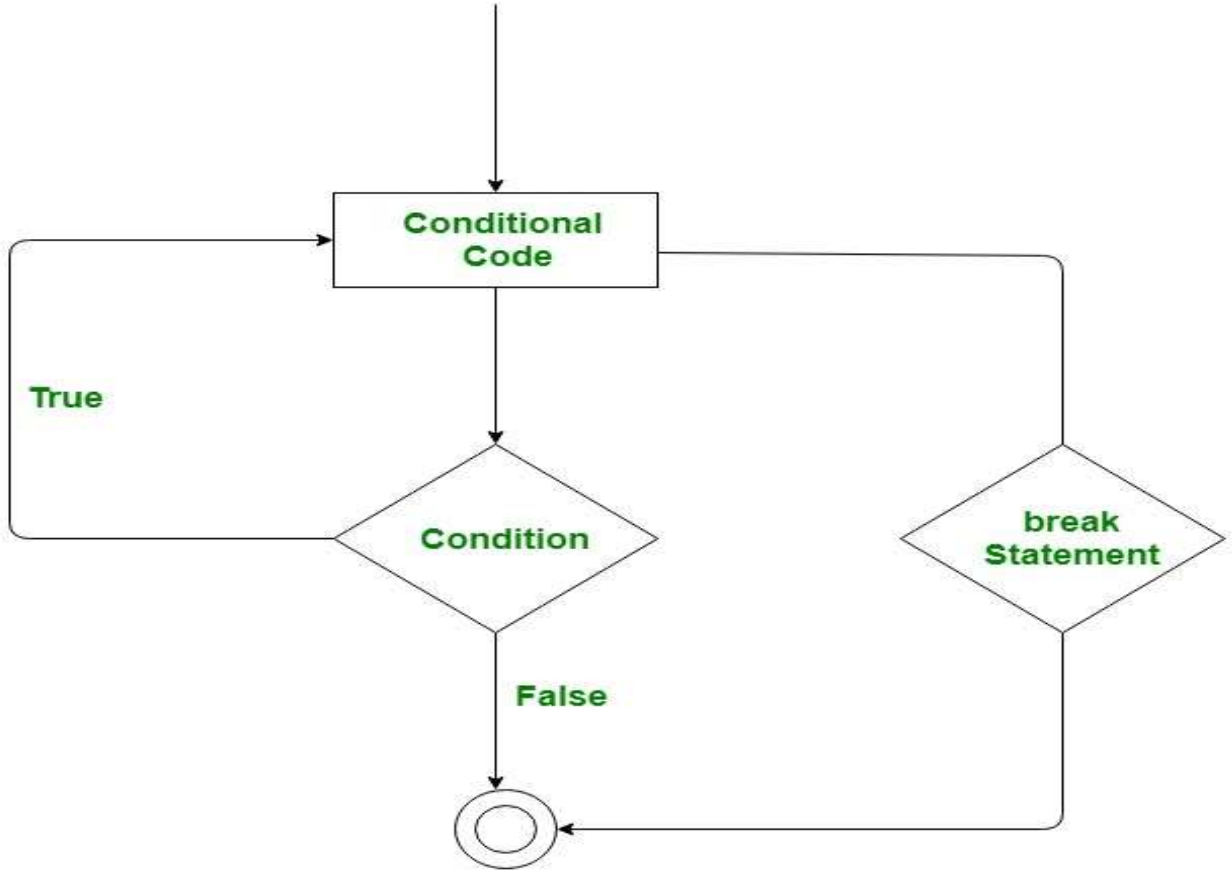
They are of four types:-

- **Break**
- **Continue**
- **goto**
- **Return**

Loops Definition & Syntax

Statements	Definition & Syntax
Break	A break statement is used to terminate the execution of the rest of the block where it is present and takes the control out of the block to the next statement. Syntax: break;
Continue	Continue statement like any other jump statements interrupts or changes the flow of control during the execution of a program. Syntax: continue;
goto	This jump statement is used to transfer the flow of control to labeled statement in the program. Syntax: goto <label>; This *label* indicates the location in the program where the control jumps to.
Return	This jump statement is usually used at the end of a function to end or terminate it with or without a value. Syntax: return<expression>;

The break statement is used to ter minate the loop or statement in wh ich it present. After that, the contro l will pass to the statements that pr esent after the break statement, if available. If the break statement pr esent in the nested loop, then it ter minates only those loops which co ntains break statement.



Flowchart of break statement

Example

Example 1: Program to print first 15 natural numbers

```
#include <stdio.h>
int main()
{
int i;
for(i=1;i<=15;i++)
{
printf("%d\n",i);
if(i==10)
break;
}
return 0;
}
```

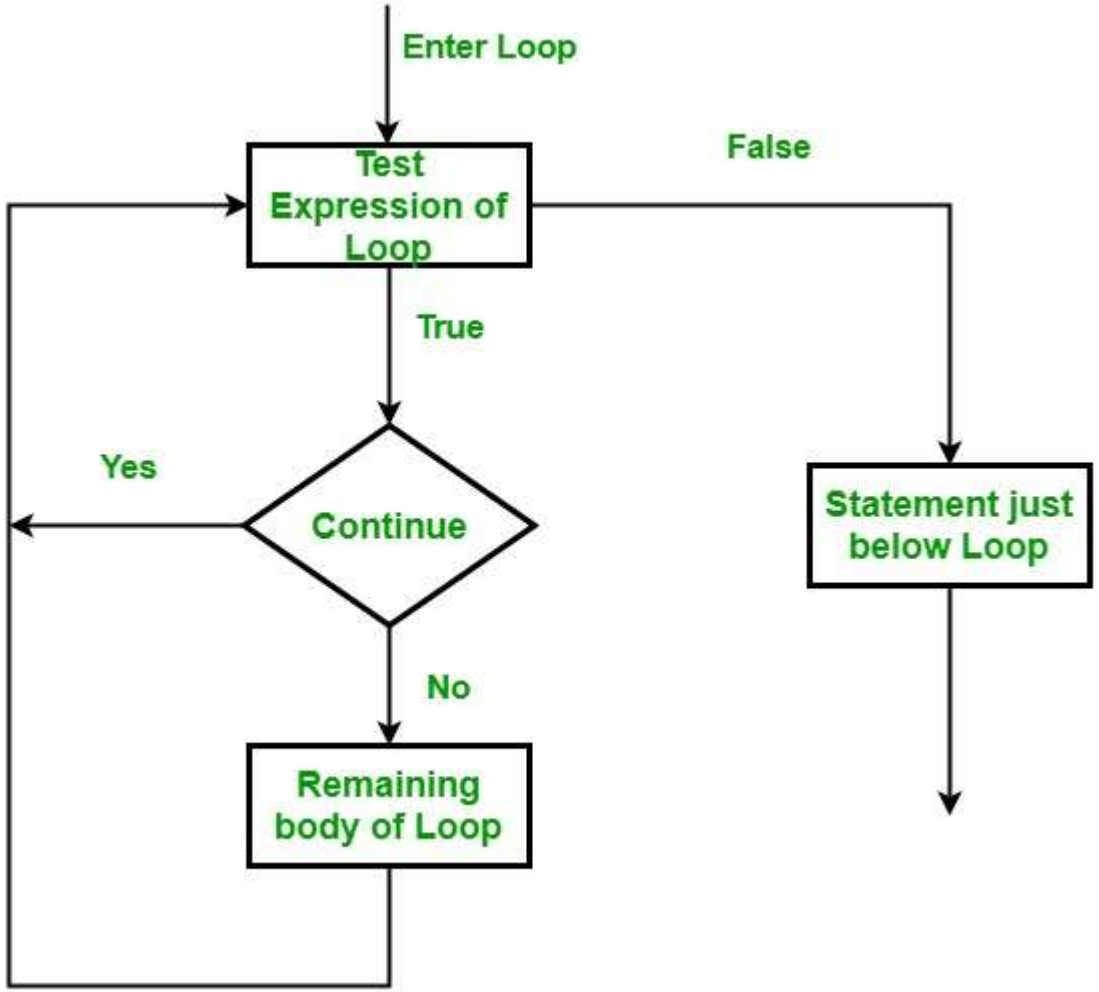
Output

```
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
1
2
3
4
5
6
7
8
9
10
```

1
1

Continue statement

This statement is used to skip over the execution part of the loop on a certain condition. After that, it transfers the control to the beginning of the loop. Basically, it skips its following statements and continues with the next iteration of the loop.



Flowchart of continue statement

Example

Write a c program to print numbers

```
#include <stdio.h>
int main()
{
int i,j;
for(i=1;i<3;i++)
{
for(j=1;j<5;j++)
{
if(j==2)
continue;
printf("%d\n",j);
}
}
return 0;
}
```

In this program, we see that the *printf()* instruction for the condition **j=2** is skipped each time during the execution because of **continue**. We also see that only the condition **j=2** gets affected by the **continue**. The outer loop runs without any disruption in its iteration.

Output

```
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
1
3
4
1
3
4
```

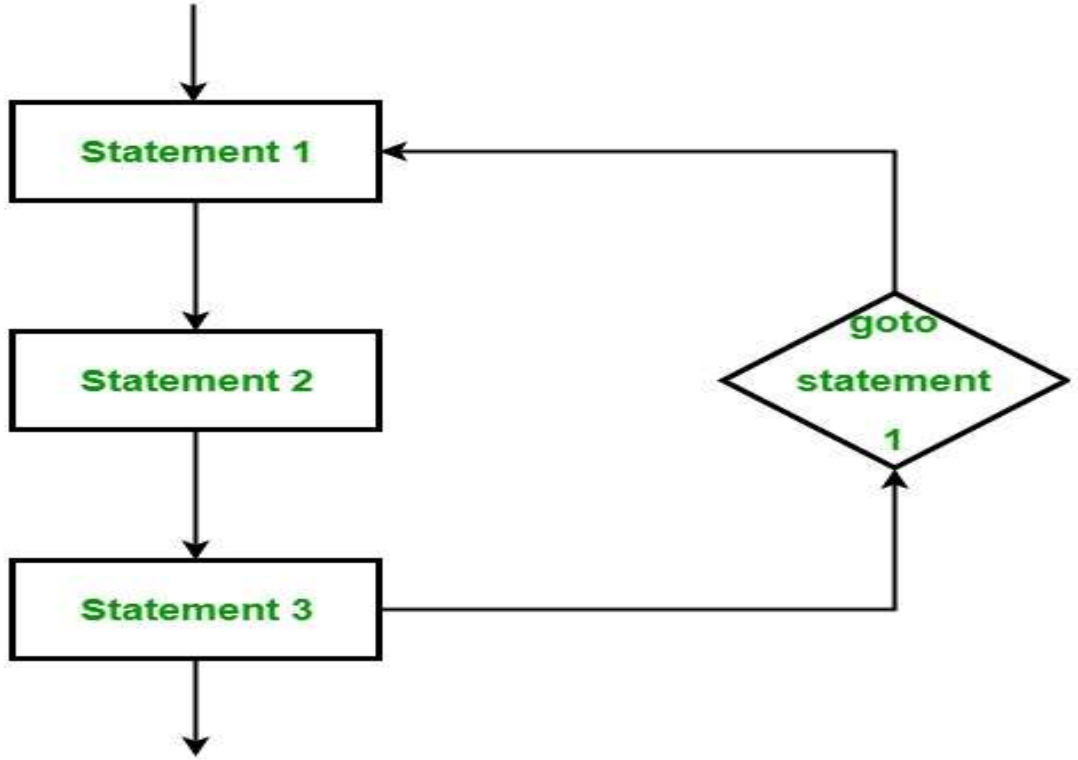
goto statement

This statement is used to transfer control to the labeled statement in the program. The label is the valid identifier and placed just before the statement from where the control is transferred.

Example:

```
#include <stdio.h>
int main()
{
int i,j;
for(i=1;i<5;i++)
{
if(i==2)
goto there;
printf("%d\n",i);
}
there:
printf("Two");
return 0;
}
```

Flowchart of continue statement




```
Output
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
1
Two
```

In this program, we see that when the control goes to the *goto there*; statement when **i** becomes equal to **2** then the control next goes out of the loop to the *label*(*there*:) and prints **Two**.

Return statement

This statement terminates the execution of the method a nd returns the control to the calling method. It returns an optional value. If the type of method is void, then the retu rn statement can be excluded.

Example:

```
#include <stdio.h>
char func(int ascii)
{
return ((char)ascii);
}
int main()
{
int ascii;
char ch;
printf("Enter any ascii value in decimal: \n");
scanf("%d",&ascii);
ch=func(ascii);
printf("The character is : %c",ch);
return 0;
}
```

Output

```
gcc -o /tmp/L2JPWfpmxw.o /tmp/L2JPWfpmxw.c -lm
/tmp/L2JPWfpmxw.o
Enter any ascii value in decimal: 80
The character is : P
```

In this program we have two functions that have a **return** type but only one function is returning a value [*func*()] and the other is just used to terminate the function[*main*()].

The function *func*() is returning the character value of the given number(here **110**). We also see that return type of *func*() is char because it is returning a character value.

The return in *main*() function returns zero because it is necessary to have a **return** value here because main has been given the return type int.

break	It terminate the loop or statement in which it present.
continue	It is used to skip over the execution part of the loop on a certain condition.
Goto	It transfer control to the labeled statement in the program.
return	It terminates the execution of the method and returns the control to the calling method.

FAQ

Q1 Write a program in C to display the sum of the numbers enter by a user and stop taking input when user enter zero.

```
 #include<stdio.h>
 void main()
{
 int num, sum=0, i,n;
 printf("Enter Number of inputs\n");
 scanf("%d",&n);
 for(i=1;i<=n;++i)
 {
 printf("Enter num%d: ",i);
 scanf("%d",&num);
 if(num==0)
 {
 break;      /*this breaks loop if num == 0 */
 printf("Loop Breaked\n");
 }
 sum=sum+num;
 }
printf("Total is %d",sum);
 getch();}
```

Output

```
gcc -o /tmp/JfA6xEVTs5.o /tmp/JfA6xEVTs5.c -lm
/tmp/JfA6xEVTs5.o
Enter Number of inputs
10
Enter num1: 5
Enter num2: 25
Enter num3: 40
Enter num4: 60
Enter num5: 0
Total is 130
```

- **Q2 Write a program to check if a number is even or not and print using the goto statement.**

```
#include <stdio.h>
int main()
{
int num;
printf("Enter a number: ");
 scanf("%d",&num);
if (num % 2==0)
goto even;
else
goto odd;
even:
printf("even Number");
Odd:
printf("odd Number");
return 0;
}
```

Assessment Questions

1.
```
#include<stdio.h>
int main()
{
int x;
for(x=1;x<=10;x++)
{
    if(x%2==1)
    continue;
    printf("\t%d",x);
}
return 0;
}
```

2.
```
#include<stdio.h>
int main()
{
int x=1;
for(;;)
{
   if(x>5)
      break;
   printf("\t%d",x++);
}
return 0;
}
```

3. The break statement is used in?

a) for loop
b)switch statement
c) while loop
d)None of the above

4.
```
#include <stdio.h>"
int main()
{
  int i;
  goto LOOP;
  for (i = 0 ; i < 10 ; i++)
  {
     printf("GeeksQuizn");
     LOOP:
      break;
  }
  return 0;
```

References

Book References:
https://www.pdfdrive.com/learn-to-program-with-c-learn-to-program-using-the-popular-c-programming-language-e166650744.htmlhttp://www2.cs.uregina.ca/~hilder/cs833/Other%20Reference%20Materials/The%20C%20Programming%20Language.pdf
http://www.freebookcentre.net/programming-books-download/The-Basics-of-C-Programming.html

Vedio Lecture: http://www.digimat.in/nptel/courses/video/106104128/L17.html
https://spoken-tutorial.org/watch/C+and+Cpp/Loops/English/
https://www.youtube.com/watch?v=67TlTBT68LE
https://www.youtube.com/watch?v=Bv1LcqhqnZs

Websites: https://www.cs.auckland.ac.nz/references/unix/digital/AQTLTBTE/DOCU_075.HTM
https://www.programiz.com/c-programming/c-goto-statement
https://www.geeksforgeeks.org/c-sharp-jump-statements-break-continue-goto-return-and-throw/

THANK YOU