**Experiment: 4**

PART A

(PART A: TO BE REFERRED BY STUDENTS)

**Aim:** Programming using looping and unconditional statements

**Learning Outcomes:** The learner would be able to

1. Understand the syntax of looping statements
2. Use looping to solve problems by writing programs
3. Work with unconditional (break/continue) statements

**Theory:**

**C++ Control Constructs/Structure/Statements**

* Control statements are used to alter the flow of program execution.
* Control statements evaluate the condition (uses relational and/or logical operators) & control the flow of execution.
* C++ control constructs/statements are as follows.

|  |  |  |
| --- | --- | --- |
| **Decision Making Statements**  **or**  **Conditional Statements**  **Or**  **Selection Statements** | **Loop Control Statements**  **or**  **Iterative Statements** | **Jump Control Instructions or**  **Branching Statement** |
| * if | * for | * break |
| * if-else | * while | * continue |
| * Nested if-else | * do-while | * return |
| * else if Ladder |  |  |
| * switch-case |  |  |

**Loop Control Statements Or Iterative Statements**

**Loop???**

Statements in a block are repeatedly executed for a certain number of times or periods.

**What are the steps used in the loop**?

1. Initialization
2. Condition or Test Expression
3. Update expression.

**Initialization: -**

* Initial value, i.e., starting value, is assigned to loop variable
* Executed only ones in the lifetime of the loop.

**Condition: -**

* Every time the condition is evaluated, if it is evaluated as true, control will get entry in the loop; otherwise, the loop will be terminated.
* Condition is true or false as we use relational and/or logical operators to write condition.

**Update Expression: -**

* Any expression to update the loop variable, after updating the loop variable control will recheck the condition & this will be iterative till the condition becomes false.

**Loops are categorized into**

1. Entry Controlled Loop
2. for
3. while
4. Exit Controlled Loop
5. do while

**for loop**

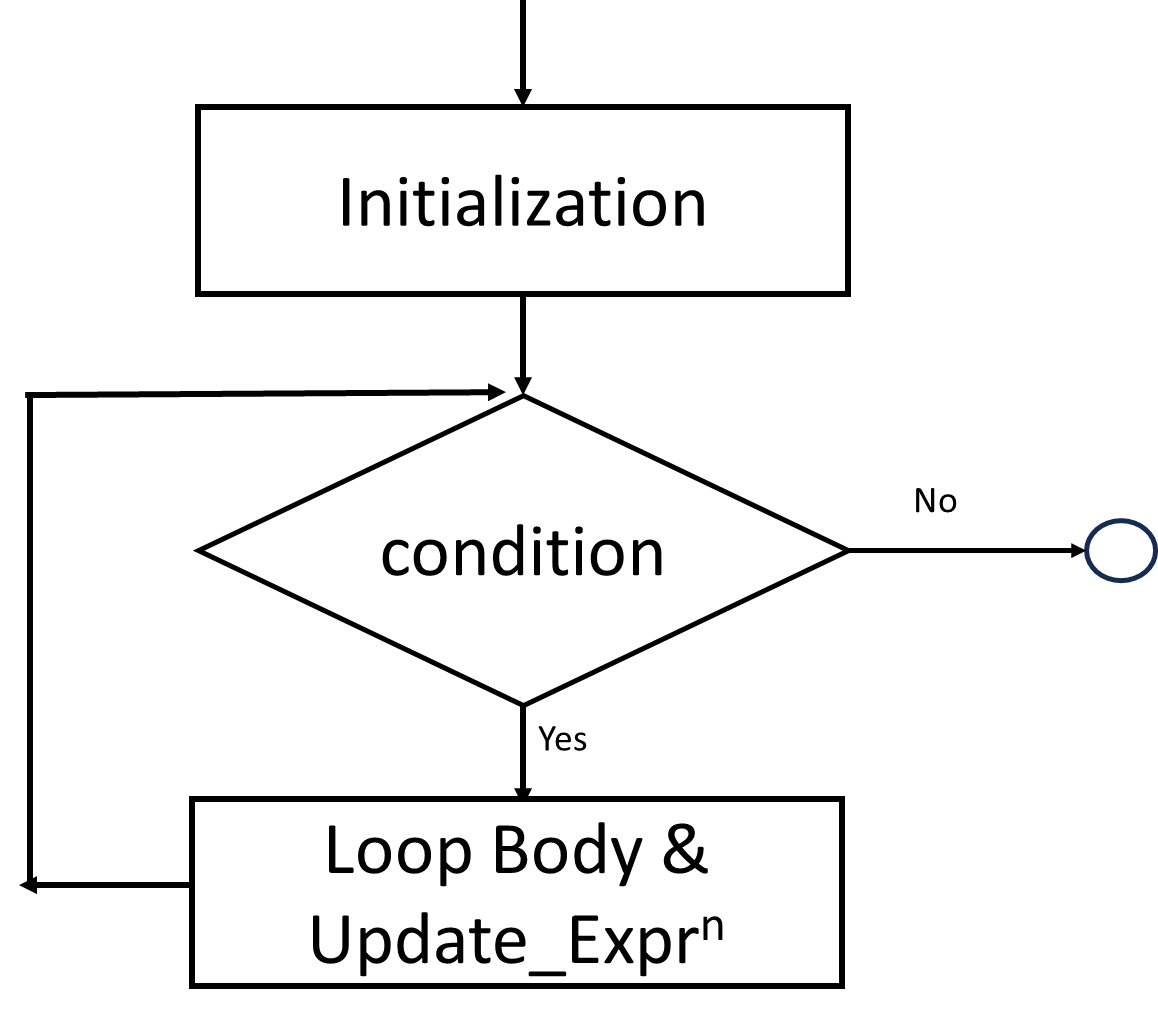
* it is iterative or loop controlled statement.
* **for** is keyword used as loop control statement.
* **Syntax:-**

for( initialization ; condition ; update\_expression ){

// Loop Body; //body of for

}

* **Flowchart:-**



**Example:- Programming example to show use of for loop.**

/\* Write a c++ program to print first ‘n’ numbers. \*/

#include<stdio.h>

int main( ){

int i,n;

cout<<“Enter value of n:”;

cin>>n;

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

cout<< “ ”<<i;

}

**Output**

Enter value of n:8

1 2 3 4 5 6 7 8

**/\* Write a c++ program to print sum of first n numbers. \*/**

/\* i. e. sum=1+2+3+…………+n\*/

#include<stdio.h>

int main( ){

int i,n,sum=0;

cout<< “Enter value of n:”;

cin>>n;

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

sum+=i;

cout<< “ Sum of numbers is ”<<sum;

return 0;

}

**Output**

Enter value of n:8

1 2 3 4 5 6 7 8

Sum of numbers is 36

**While loop**

* it is loop control statement or iterative statement.
* while is keyword & used as loop control statement.

**Syntax:-**

initlialization;

while(condition){

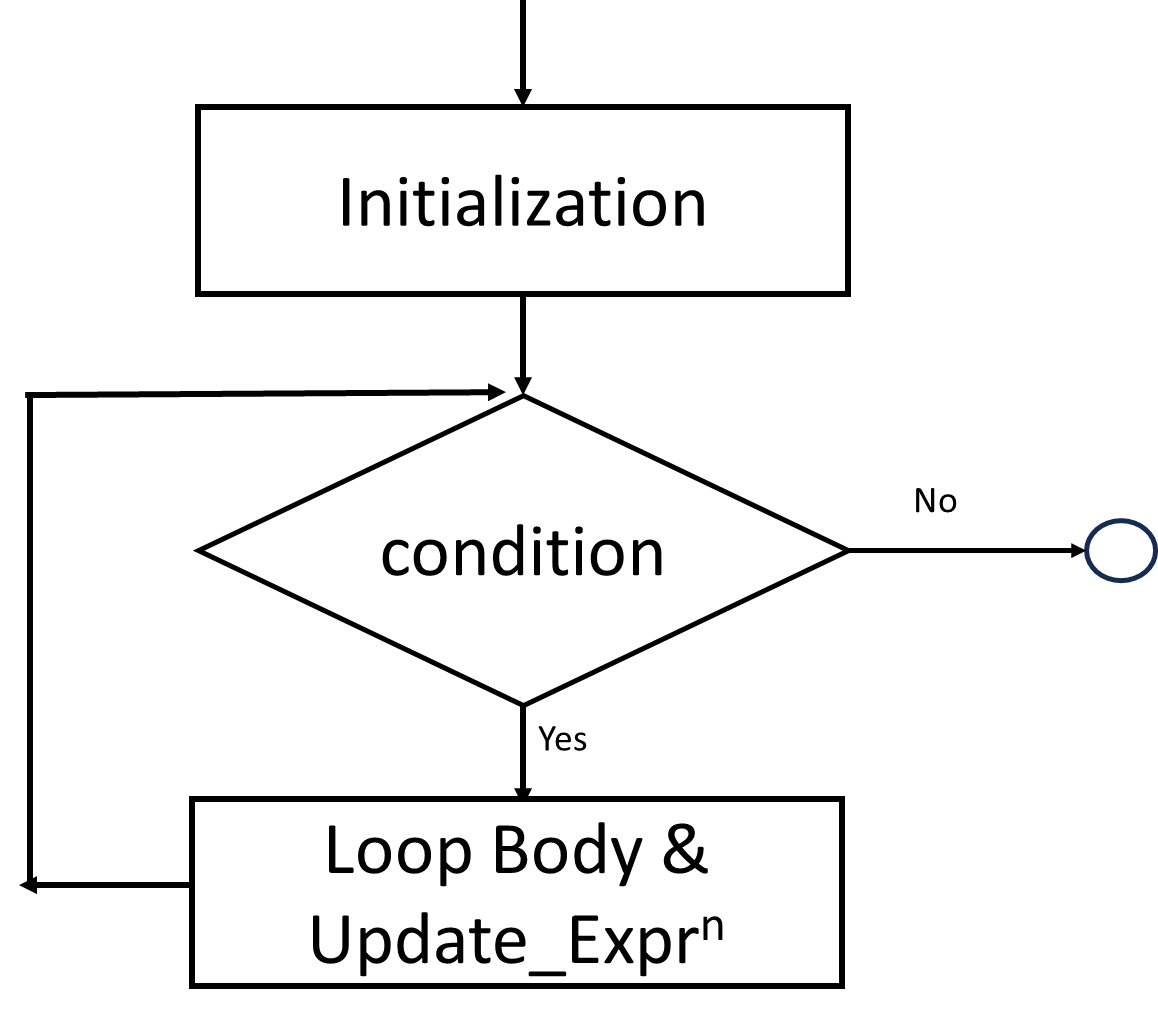
//loop body

update\_expression;

}

* In the above syntax, the condition is evaluated first; if it is true, then the loop's body is executed along with update\_expression. Again, the condition will be checked & if it is true, then the body of the loop is executed again. Otherwise, execution will be terminated.

**Flowchart: -**



**Example:- Programming example to show use of while loop.**

/\* Write a c++ program to print first ‘n’ numbers. \*/

#include<stdio.h>

int main( ){

int i,n;

cout<<“Enter value of n:”;

cin>>n;

i=1; //initial value of loop variable…

while(i<=n){

cout<< “ ”<<i;

i++; //update expression to update loop variable…

}

}

**Output**

Enter value of n:9

1 2 3 4 5 6 7 8 9

**/\* Write a c++ program to print reverse of a given number. \*/**

int main( ){

int n,d,rev=0;

cout<<“Enter one number\n”;

cin>>n;

while(n!=0){

d=n%10;

rev=rev\*10+d;

n/=10; **//** Update expression.

}

cout<< “Reverse number is ”<<rev;

return 0;

}

**Output**

Enter one number

2357

Reverse number is 7532

**do-while**

* it is exit controlled loop
* it is loop control statement or iterative statement.
* do & while are keywords & used as loop control statement.

**Syntax:-**

Initlialization;

do{

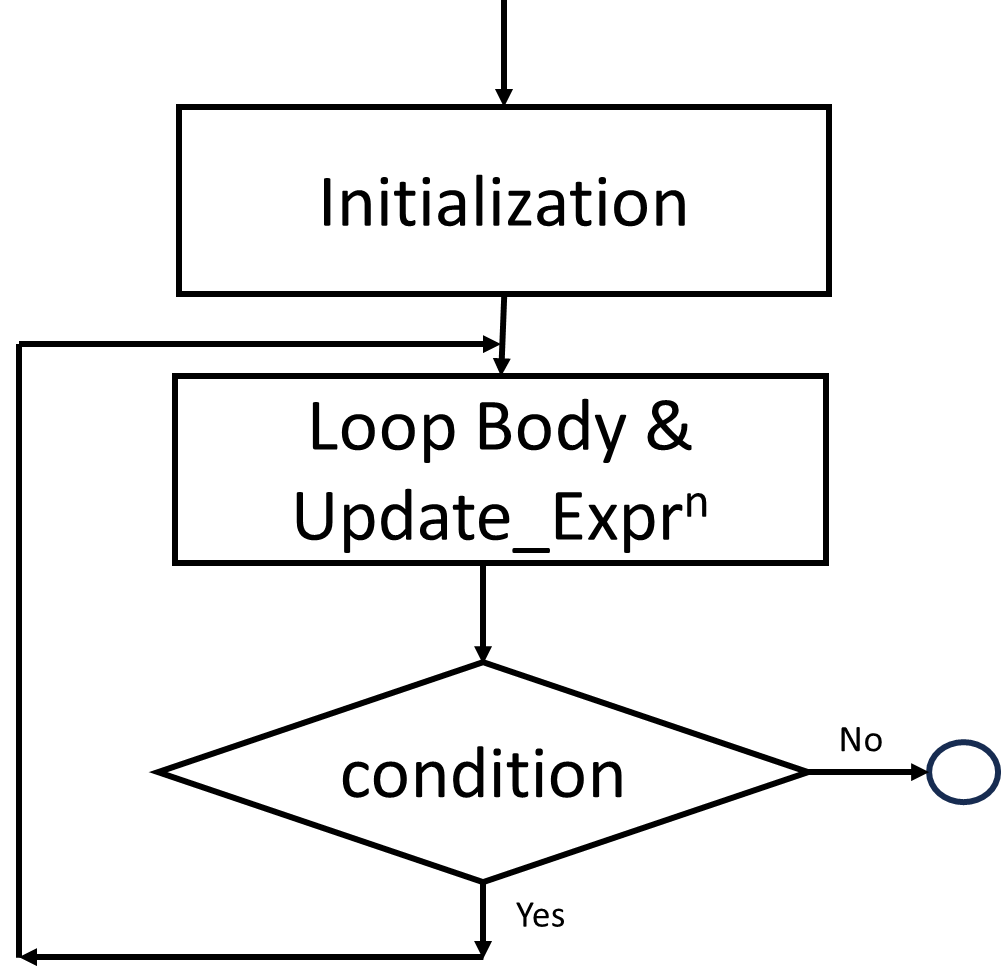
//loop body

update\_expression;

} while(condition);

* In the above syntax, the loop is executed the first time before checking the condition; after that, the condition is evaluated for every iteration; if it is evaluated as true, then the body of the loop is executed along with update\_expression. Otherwise, execution will be terminated.

**Flowchart:-**

****

**Example:- Programming example to show use of do-while loop.**

/\* Write a c program to print first ‘n’ numbers. \*/

#include<stdio.h>

int main( ){

int i,n;

cout<<“Enter value of n:”;

cin>>n;

i=1; //initial value of loop variable…

do {

cout<< “ ”<<i;

i++; //update expression to update loop variable…

} while(i<=n);

}

**Output**

Enter value of n:9

1 2 3 4 5 6 7 8 9

**Write a program that will calculate the sum of every third integer beginning with i=2 and for all values of i that are less than 100 using i. for ii while iii. Do while loop.**

|  |  |  |
| --- | --- | --- |
| **Using For** | **Using While** | **Using Do While** |
| int main( ){  int i , sum=0;  for(i=2; i<100; i=i+3){  sum = sum + i;  }  cout<<“Sum is”<<sum;  return 0;  } | int main( ){  int i=2 , sum=0;  while(i<100){  sum = sum + i;  i = i + 3;  }  cout<<“Sum is”<<sum;  return 0;    } | int main( ){  int i=2 , sum=0;  do{  sum = sum + i;  i = i + 3;  } while(i<100);  cout<<“Sum is”<<sum;  return 0;  } |

**Jump Control Instructions or Branching Statement**

* The following are the jump control or branching statements supported by C++.
  + break
  + continue
* Jump control statement transfers the control from one position to another in the program during execution.

**break:-**

* ‘break’ is used to exit the execution of any loop.
* ‘break’ is used to stop the execution of remaining cases in the switch.

**Syntax:-**

                 break;

* + In the above syntax, we have only break keywords followed by semicolon, and we can write this in switch and any loops.

**Example:- Programming example to show use break.**

|  |  |
| --- | --- |
| #include<stdio.h>  main(){  int i;  i=1;  while(1){  if(i==4) break;  cout<<“\t”<<i;  i++;  }  }  **Output**  1 2 3 4 | In this example, we have used the ‘break’ keyword to break the while loop… and execution of the while loop will be terminated whenever condition if(i==4) is evaluated as true, as the break will get executed. |

**Continue:-**

* ‘continue’ is used to continue the execution of the next iteration by skipping the current iteration of any loop.
* Note:- continue can not be used in switch-case.

* **Syntax:-**

                  continue;

* + In the above syntax, we have only the ‘continue’ keyword followed by a semicolon and we can write this in any loop.

**Example:- Programming example to show use continue.**

|  |  |
| --- | --- |
| #include<stdio.h>  main( ){  int i;  for(i=1;i<=5;i++){  if(i==4) continue;  cout<<“\t”<<i;  }  }  **Output**  1 2 3 5 | In this example, we have used the ‘continue’ keyword to continue the iteration of the loop. Whenever condition if(i==4) is evaluated as true, then the next iteration will be continued. |

**Instructions: - All the students are informed to write all executed code in a workbook in the following sequence and format.**

* + - 1. Problem Statement
      2. Input and Output
      3. Test Cases
      4. Flowchart
      5. Program (with color codes)

1. Red – Directives
2. Blue – Keywords, constants values
3. Green – Comments, messages
4. Black – {variables, functions, class, object} name, operators, punctuation
   * + 1. Trace Table (additional columns may require in some concepts)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Var-1 | Var-2 | …. | Var-n | Condition | Output |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Tasks:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No. | Problem Statement | I/O | Test Cases | Flow chart | Program- with color codes | Trace Table |
| 1 | Write a program to display the sum of N terms of even natural numbers.  Hint:-Suppose value of N=6, then first N terms are 2+4+6+8+10+12 | ✓ | ✓ |  | ✓ |  |
| 2 | Write a program to find the sum of all numbers between M and N, where N>M. | ✓ | ✓ |  | ✓ |  |
| 3 | Write a program to find the power of a number XY;here, X is base and Y is exponent. | ✓ | ✓ | ✓ | ✓ | ✓ |
| 4 | Write a program to count +ve number, -ve number and zeros until user want, make use of do while loop. | ✓ |  |  | ✓ |  |
| 5 | Write a program to check whether a number is a strong number or not. | ✓ |  |  | ✓ |  |
| 6 | Write a program to accept a number from the user. Find and print the sum of digits of the number. | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7 | Write a program to accept a number from user and display if the number is Armstrong number. (Armstrong number is the number in any given number base, which forms the total of the same number, when each of its digits is raised to the power of the number of digits in the number.) | ✓ |  | ✓ | ✓ | ✓ |
| 8 | Write a program to print the sum of the last and the first digit of a number the user gives. | ✓ |  |  | ✓ |  |
| 9 | Write a program to check whether the entered number is a palindrome. | ✓ | ✓ | ✓ | ✓ | ✓ |
| 10 | Implement a program to print all Leap Years from 1 to N using C++ program. | ✓ |  |  | ✓ |  |
| 11 | Write a program to check whether the entered number is prime or not. (make use of **break**) | ✓ |  | ✓ | ✓ |  |
| 12 | Write a program to print the entire uppercase and lowercase letters using a loop (use **continue**). | ✓ |  |  | ✓ |  |