Introduction to Computing and Programming

Loops

Recap

BITWISE OPERATORS

Example: Operators Precedence & Associativity

Type Conversion

Conditional Statements/Decision making in C

Contents

Type Conversion in C

Data type conversion is required when dissimilar data types appear in an expression.

Types of conversion:

- Implicit type conversion: Compiler does the conversion on its own so that the data types are compatible with each other.
- Explicit type conversion: Compiler forcefully performs the conversion, which is carried out by the type cast operator.

Decision making statement

- if Statement
- if-else Statement
- if-else-if Ladder
- Nested if Statement
- switch Statement
- Conditional Operator
- Jump Statements:
  - Break
  - continue
  - goto
  - return

Conditional Statement In C

If-else

If

```
if(condition)
{
   //true
}
```

If-else

```
if(condition)
{
   //true
} else {
   //false
}
```

If-Else If

```
if(condition 1 )
{
   //true
} else if (condition 2)
{
   //true
} else {
   //false
}
```

Nested If-Else

```
if(condition 1)
{
   //true
} if (condition 2){
   //true
} else {
   // 1 is true not 2
} else {
   // 1 is false
}
```

Switch

```
switch(expression)
{
      case 1:
      break;
      case 2:
      break;
      case 3:
      break;
      default;
```

Write a C program to check whether a number is positive or not.

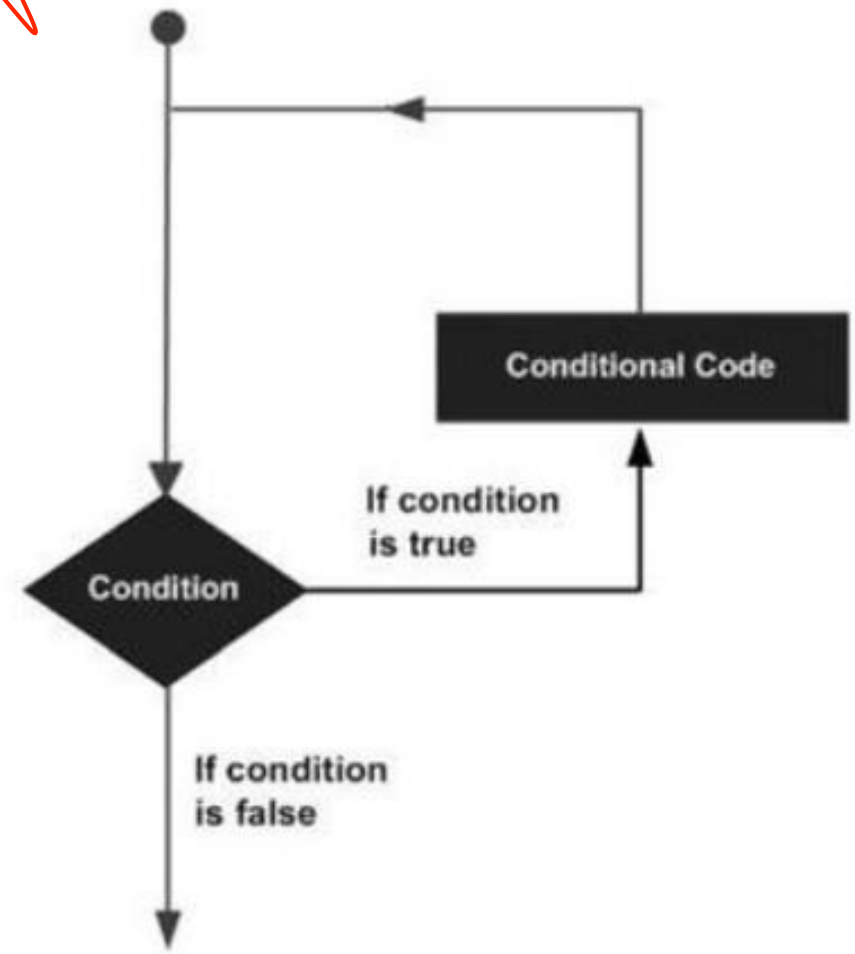
```
#include <stdio.h>

int main() {
   int num;  // Declare a variable to store the number

   // Ask the user for input
   printf("Enter a number: ");
   scanf("%d", &num);  // Read the number from the user

   // Check if the number is positive, negative, or zero
   if (num > 0) {
      printf("%d is a positive number.\n", num);
   } else if (num < 0) {
      printf("%d is a negative number.\n", num);
   } else {
      printf("The number is zero.\n");
   }

   return 0;
}
```



What is Loop?

- A loop is a sequence of instructions that is **continually repeated** until a certain condition is reached.
- They reduce the need for repetitive coding and **improve efficiency**.

Types of Loop

- There are **three types** of loops:
  - Using a while statement
  - Using a for statement
  - Using a do-while statement

Loops Description

Loop Type	Description
While loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
For loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Do……While loop	Like a while statement, except that it tests the condition at the end of the loop body

While Loop

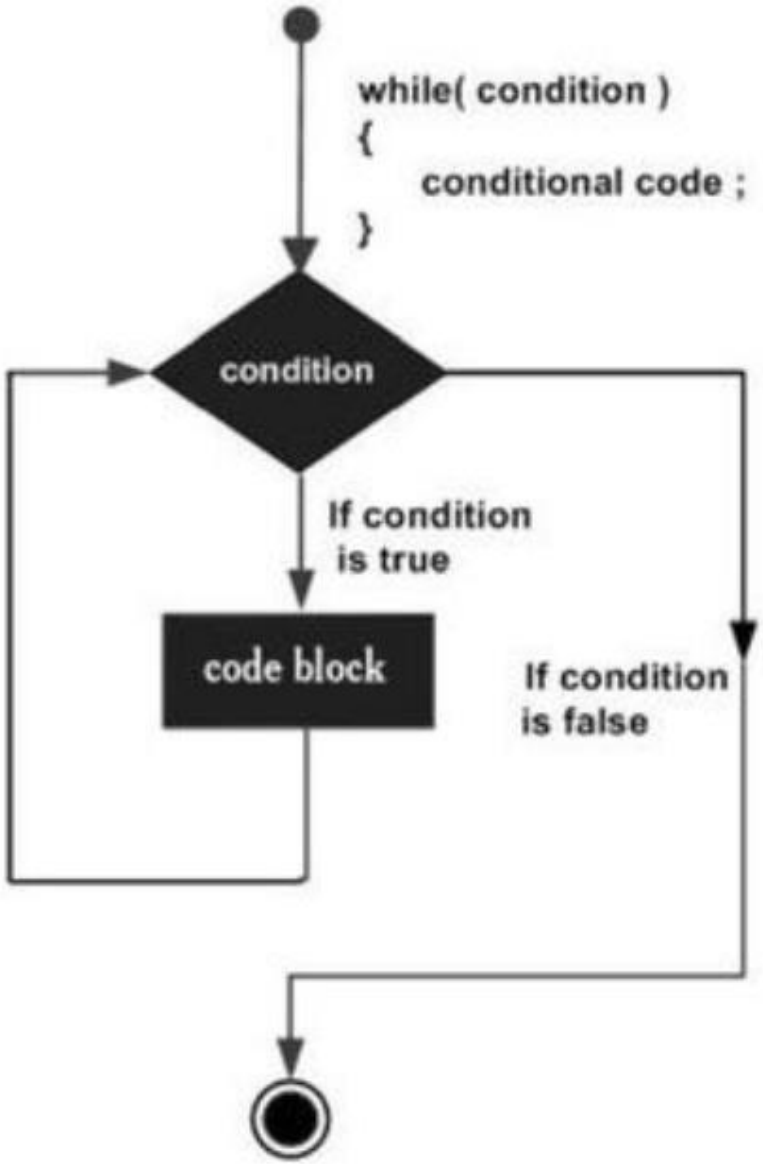
- The loops continues as long as the **condition is true**.
- If the condition is false initially, the loop may not execute at all.
- **Syntax:**

```
Initialization ;

                    while( condition )
 {
            statement (s) ;

                        Increment ;
  }
```

Flow chart of While Loop



Examples of While Loop

Eg1. Print ‘Hello world’ 10 times

```
#include <stdio.h>

int main() {
   int i = 0;  // Initialize the loop counter

   while (i < 10) {  // Loop will run as long as i is less than 10
      printf("Hello world\n");
      i++;  // Increment the loop counter
   }

   return 0;
}
```

Eg2. Print the numbers from 0 to 4 in newline

```
#include <stdio.h>

int main() {
   int i = 0;  // Initialize the loop counter
while (i < 5) {
   printf("%d\n", i);
   i++;
}
```

To count number of characters using While loop

```
#include <stdio.h>
int main() {
   int count = 0;     // Initialize counter to zero
   char ch;          // Variable to store each character input

   ch = getchar();    // Read a character from standard input
   while (ch != '\n') { // Continue looping until a newline character is encountered
      count++;       // Increment the character count
      ch = getchar(); // Read the next character
   }

   printf("You entered %d characters", count); // Output the total number of characters

   return 0;
}
```

To count number of characters using While loop (Efficient way)

```
#include <stdio.h>
int main() {
   int count = 0;     // Initialize the character counter to zero
   char ch;          // Variable to store the current character input

   ch = getchar();    // Read the first character from standard input

   while ((ch = getchar()) != '\n') { // Read and check each character until a newline is encountered
      count++;       // Increment the character count for each character read
   }

   printf("You entered %d characters", count); // Print the total number of characters entered

   return 0;
}
```

Infinite loop using While

- An **infinite while loop** is a loop that runs **indefinitely** because its condition always evaluates to true.
- **Syntax**: while (1) {

  // Code to be executed

}
- The code inside the loop executes repeatedly without end **or forever** unless its body contains a statement that transfers control out of the loop **(such as break, goto, return)**

Breaking with break;

➢ **To count number of characters**

```
#include <stdio.h>

int main() {
   int count = 0;    // Initialize the counter to zero
   char ch;        // Variable to store each character input

   while (1) {      // Infinite loop
      ch = getchar(); // Read a character from standard input
      if (ch == '\n') {  // Check if the character is a newline
         break;      // Exit the loop if newline is encountered
      }
      count++;       // Increment the character count
   }

   printf("You entered %d characters\n", count); // Print the total number of characters

   return 0;
}
```

N

For Loop

- Combines **initialization, condition-checking, and increment/decrement** in a single line.
- Commonly used when the number of iterations is known.

- **Syntax:**

```
for ( init; condition; increment )
{
statement(s);
}
```

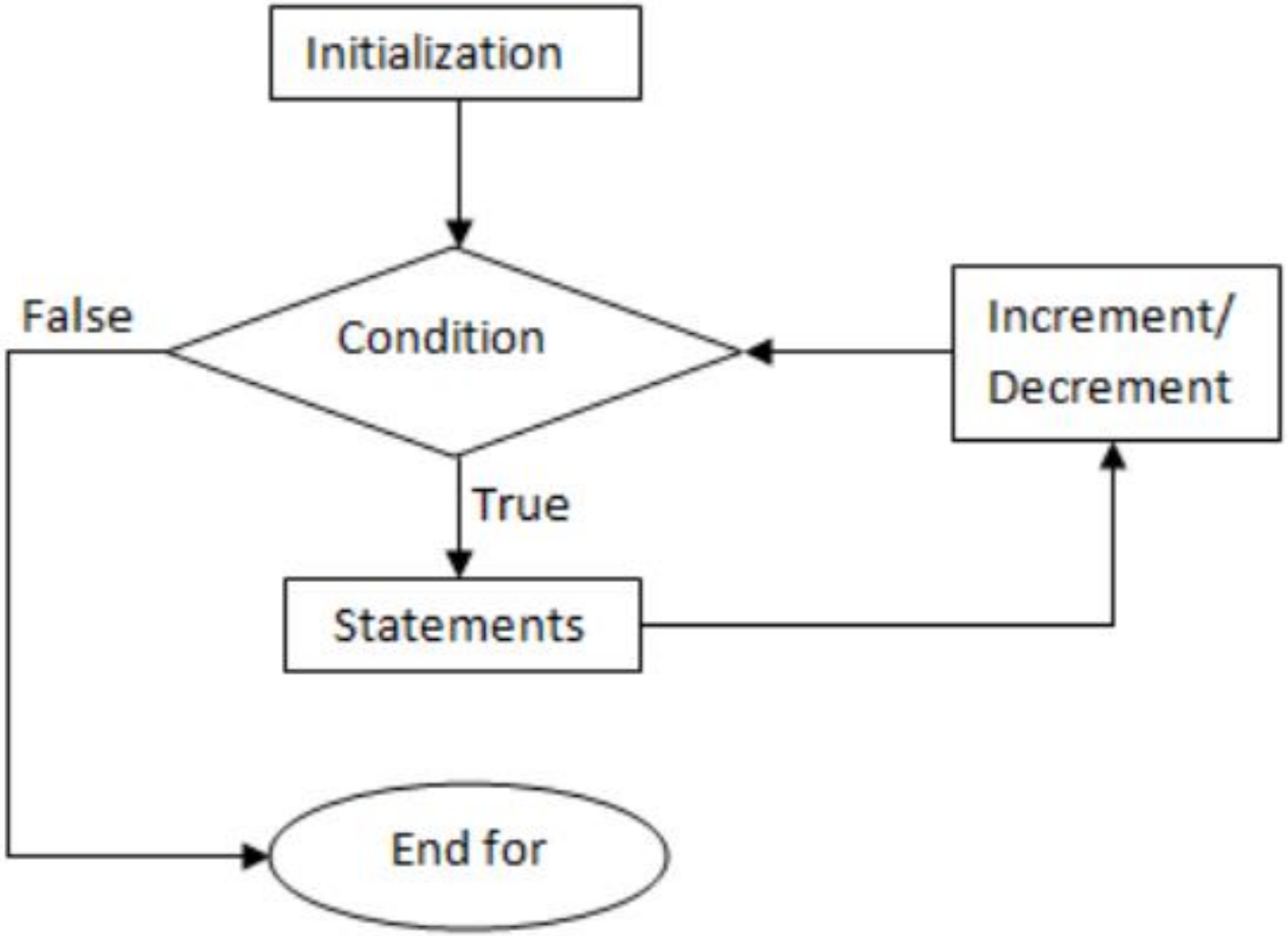
- **The for statement syntax is:**

```
for(expr1; expr2; expr3)
Statement
```

- **This is equivalent to:**

```
 expr1;
while (expr2) {
statement
expr3;
}
```

Flow chart of For Loop



Examples of for Loop

Eg1. Print ‘Hello world’ 10 times

```
#include <stdio.h>


int main() {
   // Using a for loop to print "Hello world" 10 times
   for (int i = 0; i < 10; i++) {
      printf("Hello world\n");
   }


   return 0;
}
```

Eg2. Print the numbers from 0 to 4 in newline

```
#include <stdio.h>


int main() {
   // Using a for loop to print numbers from 0 to 4, each on a new line
   for (int i = 0; i < 5; i++) {
      printf("%d\n", i);
   }


   return 0;
}
```

Infinite for loop

- There may be a condition in a for loop **which is always true**. In such case, the loop will run infinite times.
- for(int i=0;i>0;i++){ **//This is not an infinite loop**

  printf("This is an infinite loop"); }
- Or for(int i=0;;i++){

  printf("This is an infinite loop"); }
- Or for(;;){

  printf("This is an infinite loop"); }

Solution using break:

- for(;;){

  printf("This loop will only run once");

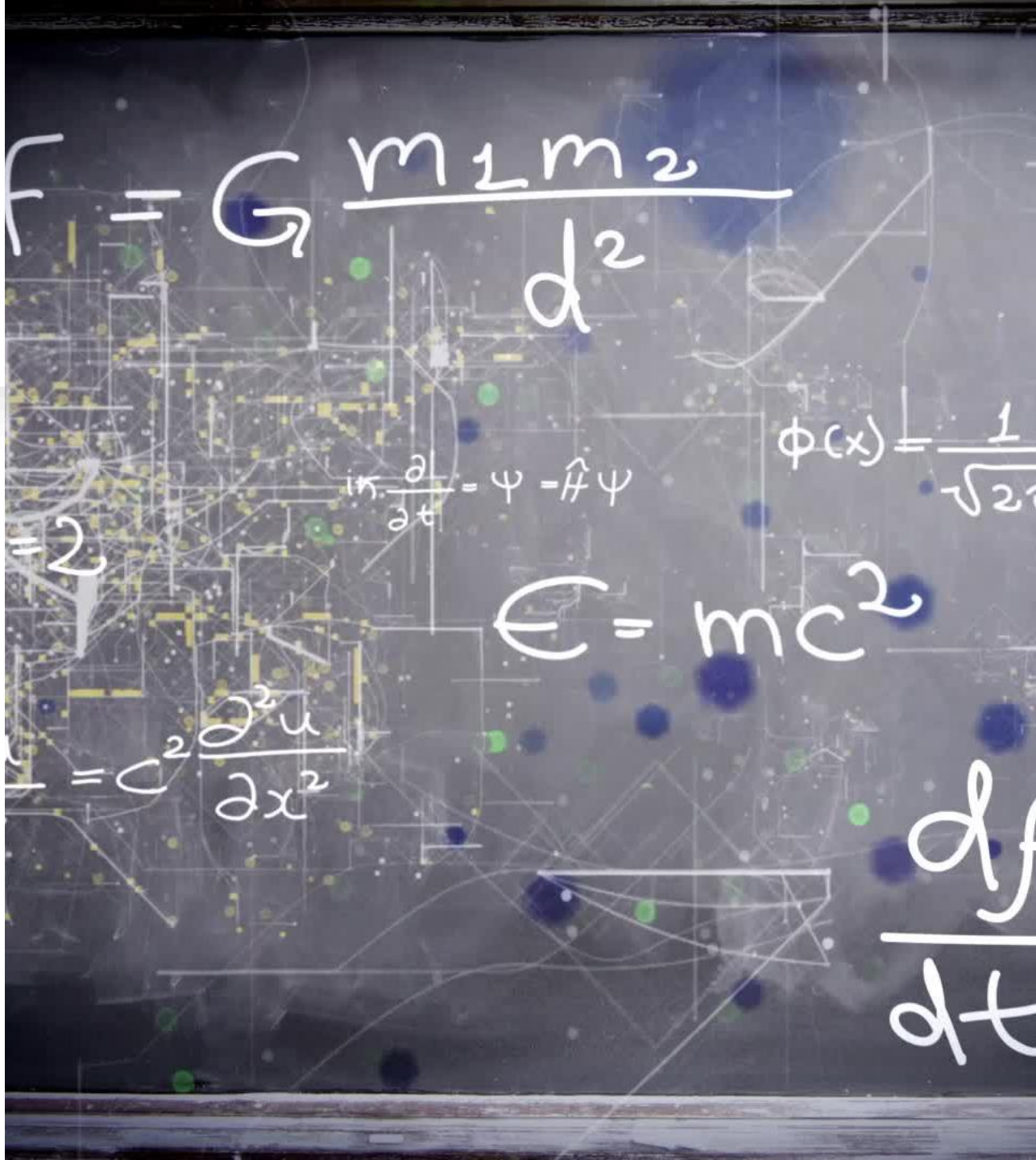
  break;}

Comma (,)

- comma can be used both as an **operator** and as a **separator.**
- int j,k,l; /* , is a separator */
- printf(“%d %d %c”, j, k, c); /* , is a separator*/
- expr1,expr2 → is an expression and has value equal to expr2
- // Using the comma operator
- **comma has precedence less than = (assignment).**
- Associativity is from **left to right**
- **x= 1,2;** // Output: x= 1
- **x = (1, 2, 3);** // x will be assigned the value of the rightmost expression, which is 3
- **y = (4, 5, 6);** // y will be assigned the value of the rightmost expression, which is 6
- printf("x = %d, y = %d\n", x, y); // Output: x = 3, y = 6

Comma Cont.. (,)

- j = (2,3); /* what is the value of j*/
- Output is 3
- j = 2,3; /* what is the value of j */
- Output is 2 because **(j=2),3;**
- j = (2,3,4,5); /* value of j ? */
- Output is 5 because j = (((2,3),4),5)
- **comma is often used with for loops**



for loop Using comma

- /* a[ ] is an array of 10 elements */

```
for(j=0, k=9; j < k; j++,k--) {
//code for swapping two numbers
    t = a[j];
    a[j] = a[k];
    a[k] = t;
    }
```

- This will reverse the order of elements in the array a[]

for loop Using comma

//Comma as a separator in for loop

```
#include <stdio.h>

int main() {
   // Using the comma as a separator to initialize and increment multiple variables
   for (int i = 0, j = 5; i < j; i++, j--) {
      printf("i = %d, j = %d\n", i, j);
   }
   return 0;
}
```

Output: i = 0, j = 5

i = 1, j = 4

i = 2, j = 3

//Comma as a operator in for loop

```
#include <stdio.h>

int main() {
   int x;
   for (x = (1, 2); x < 3; x++) {
      printf("x = %d\n", x);  // x is initialized to 2 due to the comma operator
   }
   return 0;
}
```

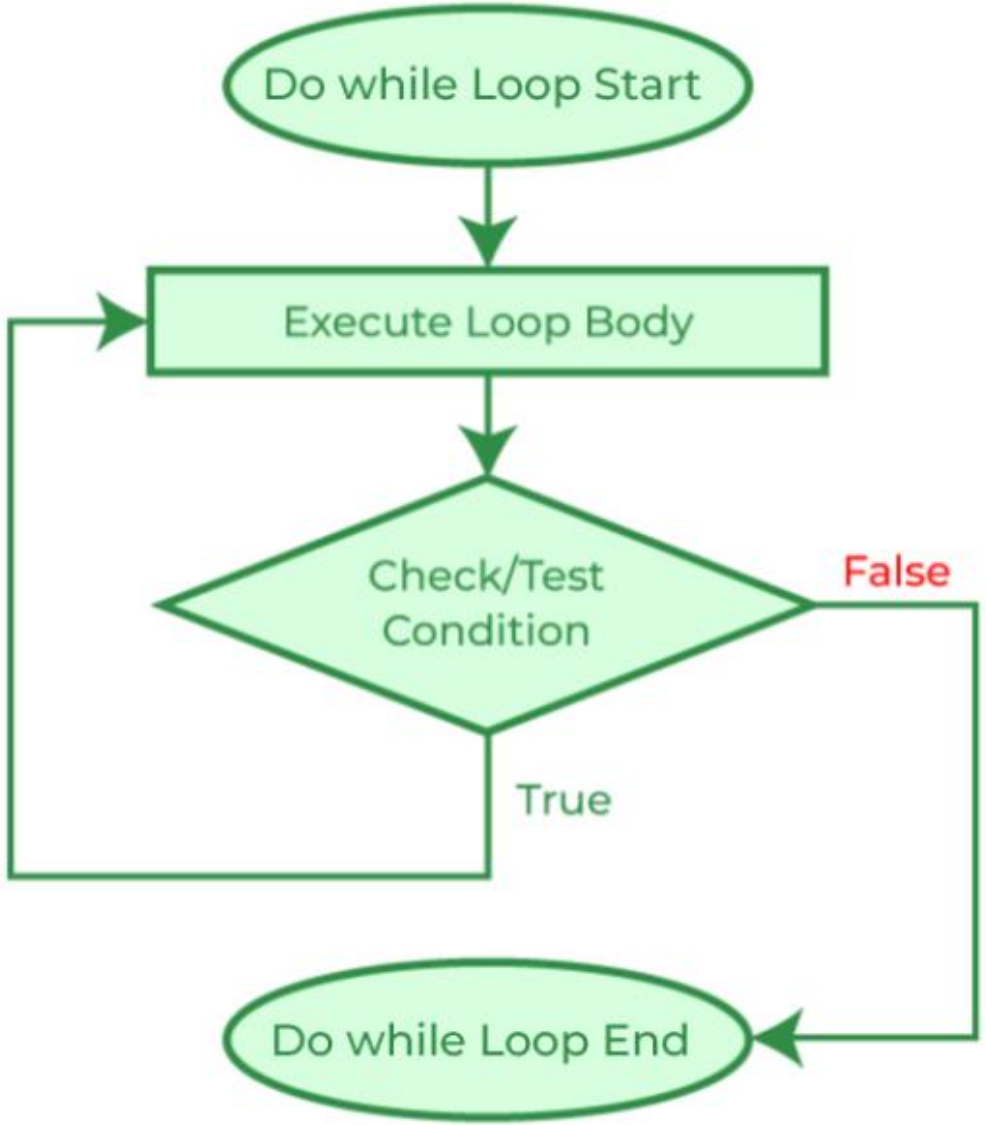
do-while Loop

- The code block is **executed at least once**, even if the condition is false.
- The condition is checked after the loop body has executed.
- The statement is executed, then expression is evaluated. If it is true, statement is evaluated again, and so on.
- When the expression becomes false the loop terminates.
- **A do-while loop is similar to a while loop, except that a do-while loop is guarantee to execute at least one time.**

- **Syntax:**

```
do
{
  // body of do...while loop
}
while (condition);
```

Flow chart of do-while Loop



Examples of do-while Loop

Eg1. Print ‘Hello world’ 10 times

```
#include <stdio.h>

int main() {
   int i = 0;  // Initialize the counter

   // do-while loop to print "Hello world" 10 times
   do {
      printf("Hello world\n");  // Print "Hello world"
      i++;  // Increment the counter
   } while (i < 10);  // Continue the loop until i is less than 10

   return 0;
}
```

Eg2. Print the numbers from 0 to 4 in newline

```
#include <stdio.h>

int main() {
   int i = 0;  // Initialize the counter

   // do-while loop to print numbers from 0 to 4
   do {
      printf("%d\n", i);  // Print the current value of i
      i++;  // Increment the counter
   } while (i < 5);  // Continue the loop until i is less than 5

   return 0;
}
```

N

Break and Continue

- **Break:** Exits the loop immediately.
- Eg:
```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
    printf("%d\n", i);
}
```
- **Output:**
```
0
1
2
3
4
```

- **continue:** Skips the remaining code in the current iteration and jumps to the next iteration.
- Eg:
```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        continue;
    }
    printf("%d\n", i);
}
```
- **Output:**
```
0
1
2
3
4
6
7
8
9
```

Nested Loops

- A **loop inside another loop**.
- **Use Cases:** Working with multidimensional arrays, complex iterations.

- **Eg:**
```
for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    printf("i = %d, j = %d\n", i, j);
  }
}
```

Output:
```
i = 0, j = 0
i = 0, j = 1
i = 0, j = 2
i = 1, j = 0
i = 1, j = 1
i = 1, j = 2
i = 2, j = 0
i = 2, j = 1
i = 2, j = 2
```

Break and Continue Cont..

- In case of **nested loops** like

```
while(exp1){
while(exp2) {
statement1
beak;

}
statement2
}
```

- break; causes the **innermost loop** to be exited.

Break and Continue Cont..

- continue statement is related to break, but less often is used.
- It skips the rest in the loop and continues with next iteration of the loop.
- In case of **for loop** it skips the rest of the loop, but it does the increment step before continuing with next iteration.
- The continue statement applies only to loops, **not to switch**.

Continue Example

- This reads ten numbers but prints roots for only +ve numbers.

- for(j=0; j < 10; j++) {

```
scanf("%f", &v);
if (v < 0) continue;
else {
sv = sqrt(v); /* include<math.h> */
printf("root is %f\n", sv);

}
}
```

Loop Best Practices

- **Avoid infinite loops** by carefully managing loop conditions.
- Keep loops **simple and readable**; avoid deeply nested loops.
- Use **meaningful variable names** for loop counters.
- Optimize loop conditions to prevent unnecessary iterations.
  Example: Sum of even numbers

- **Common Pitfalls:** Off-by-one errors: Incorrect **loop boundaries**.
- **Infinite loops** due to incorrect condition updates.
- Example**While(i<10)**

```
{ printf("the value of %d", i);
}
```

Quiz 1 (12th Sept, 12:30pm to 1:30pm) Pattern

- **Total 8 to 10 Questions**
- **Marks: 15 to 20**
- **Duration: 30 to 45 minutes**
- **These types of Questions can be asked:**
- Type 1: MCQ questions:
- Type 2: Match the following
- Type 3. Point out the errors, if any in the following C statements
- Type 4: Evaluate the following expression
- Type 5: What will be output of the following programs:
- Type 6: Theory Question
- Type 7: Write the logic for the question
- Type 8: Number System Conversion

Quiz 1 Syllabus

- All the topics covered till today (10th Sept)
- **Topics:** Introduction to Basic Fundamentals of Computers, Introduction to Programming, Identifiers and Constants, Data Types, Number System, Operators, Logical Expressions, Managing input & output, Conditional statements, Decision making & Branching, Decision making & loops
- **Note: kindly refer lecture slides as well as textbooks mentioned in the lecture 1 slides for detailed theory & practice purpose.**
- I will upload the question bank today for your reference & the practice of coding.

LASC Tutor details

- Two tutors are in ICP: **Ishan Das (id996@snu.edu.in) & Harshaditya Das (hd496@snu.edu.in).**
- The schedule for Ishan's LASC classes is as follows:
  - **Monday and Friday, 5 - 6 pm, Venue: D003**

- The schedule for Harshaditya's LASC classes is as follows:
  - **Thursday, D109, 6:30-7:30**
  - **Friday, D109, 6:30-7:30**
- Additionally, this is the WhatsApp link for these classes: https://chat.whatsapp.com/KnJRiGPiYJ076LfPemGwBj

Class Representatives Discussion

- I am the **mentor of you** (First year CSE students).
- Need to select **5 to 7 students** from you
- Give your available slots (1 hr. or min 30mins) either on Sept. 20th (Fri) or Sept. 23 (Mon) **by today 5PM**. **(Email me please)**

- **Share your achievements** with me over this drive link:
- https://docs.google.com/spreadsheets/d/1zo9Pk-ngADjoLms2aqjp803qJID-BUTKkVfGhSPtQGg/edit?gid=0#gid=0

Write a C program to find sum of n natural numbers

```
#include <stdio.h>
int main() {
   int n, sum = 0;
   // Input the value of n
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   // Make sure the input is a positive integer
   if (n < 0) {
      printf("Invalid input! Please enter a positive integer.\n");
      return 0;
   }
   // Calculate the sum of first n natural numbers using a loop
   for (int i = 1; i <= n; i++) {
      sum += i;
   }
   // Output the result
   printf("Sum of the first %d natural numbers is: %d\n", n, sum);

   return 0;
}
```

Write a C program to find the table of 2

```
#include <stdio.h>

int main() {
    int i;

    // Print the multiplication table of 2
    printf("Multiplication table of 2:\n");
    for (i = 1; i <= 10; i++) {
        printf("2 x %d = %d\n", i, 2 * i);
    }

    return 0;
}
```

Write a C program to check whether a number is palindrome or not

```
#include <stdio.h>
int main() {
  int num, reversedNum = 0, remainder, originalNum;
  // Input the number from the user
  printf("Enter an integer: ");
  scanf("%d", &num);
  originalNum = num; // Store the original number to compare later
  // Reverse the digits of the number
  while (num != 0) {
    remainder = num % 10;  // Get the last digit of the number
    reversedNum = reversedNum * 10 + remainder;  // Build the reversed number
    num = num / 10;  // Remove the last digit from the original number
  }
  // Check if the original number and reversed number are the same example 121
  if (originalNum == reversedNum) {
    printf("%d is a palindrome.\n", originalNum);
  } else {
    printf("%d is not a palindrome.\n", originalNum);
  } return 0; }
```

N

Write a C program to display a pyramid

```
#include <stdio.h>


int main() {
   int rows = 5;  // Number of rows for the pyramid

   // Outer loop to handle the number of rows
   for (int i = 1; i <= rows; i++) {
      // Inner loop to print stars for each row
      for (int j = 1; j <= i; j++) {
         printf("*");
      }
      // Move to the next line after printing each row
      printf("\n");
   }

   return 0;
}
```

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

Write a C program to display a pyramid

```
#include <stdio.h>


int main() {
   int rows = 5;  // Number of rows for the pyramid


   // Outer loop to handle the number of rows
   for (int i = 1; i <= rows; i++) {
      // Inner loop to print stars for each row
      for (int j = 1; j <= i; j++) {
         printf(“j");
      }
      // Move to the next line after printing each row
      printf("\n");
   }


   return 0;
}
```

```
1
12
123
1234
12345
```

Write a C program to check whether a number is Armstrong number or not

- **Example:**
  - **153 is an Armstrong number because 1^3+5^3+3^3=153.**
  - **122 is not an Armstrong number** because 1^3+2^3+2^3= 1+8+8 = 17 which is not equal to 122.
  - 9474 = 9^4+4^4+7^4+4^4 is an Armstrong number.

Write C programs to display these patterns

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

```
1
12
123
1234
12345
```

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

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

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

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

Examples of Loop for Practice

1. Write a C program to print numbers from 1 to 100
2. Write a C program to find the table of 5
3. Write a C program to check whether a number is even or not
4. Write a C program to check whether a number is Armstrong number or not
5. Write a C program to check whether a number is prime number or not
6. Write a C program to reverse the number
7. Write a C program to display Fibonacci series
8. Write a C program to print an inverted pyramid pattern

```
54321
4321
321
21
1
```

N

Upcoming lecture

Arrays