

# *Control Statements in C*

By

**Prof. Muhammad Iqbal Bhat**

Department of Higher Education,  
Government Degree College Beerwah

Prof. M. Iqbal Bhat (JKHED)



# Control Statements in C

Control statements are essential in programming languages to control the flow of program execution.

Control statements allow programmers to make decisions, repeat code, and transfer control to different parts of the program based on certain conditions.

C language is a widely used programming language that provides several types of control statements to help programmers write efficient and effective code.

Conditional statements, also known as selection statements, are used to execute code based on certain conditions.

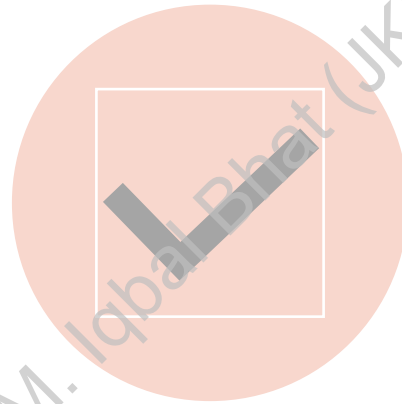
Loop statements, also known as iteration statements, are used to repeat a block of code multiple times.

Jump statements, also known as transfer statements, are used to transfer control to different parts of the program.

# Control Statements in C



CONDITIONAL  
STATEMENTS



LOOP STATEMENTS



JUMP STATEMENTS

Prof. M. Iqbal Bhat (JKHED)

# Conditional Statements:

Conditional statements are used to execute code based on certain conditions. C language provides two conditional statements: the if-else statement and the switch statement.

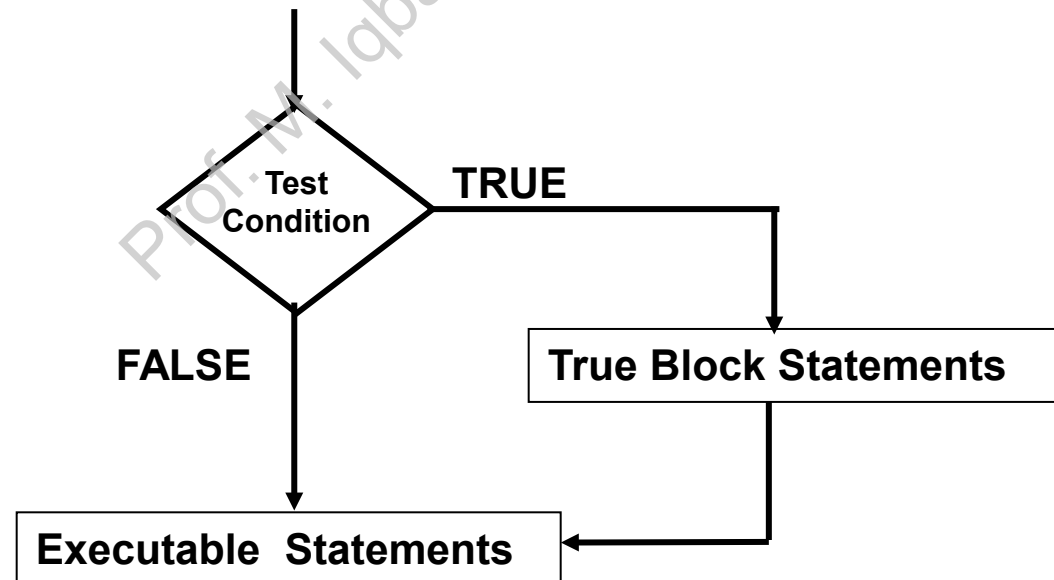
**If-else  
statement**

**Switch  
statement**

# if-else Statement:

- The if-else statement is used to execute a block of code if a certain condition is true and another block of code if the condition is false.

```
if (condition) {  
    // code to be executed if the condition is true  
}  
else {  
    // code to be executed if the condition is false  
}
```



+

•

○

# if-else Statement:

---

- Check if a number is positive or negative.

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);

    if (num >= 0) {
        printf("%d is positive.\n", num);
    }
    else {
        printf("%d is negative.\n", num);
    }
    return 0;
}
```



Prof. M. Iqbal Bhat (JKHED)

# if-else Statement: Variations

---

```
//Simple if
if (condition) {
    // code to be executed if the condition is true
}
```

```
//if-else statement:
if (condition) {
    // code to be executed if the condition is true
}
else {
    // code to be executed if the condition is false
}
```



# if-else Statement: Variations

---

```
//Nested if-else
if (condition1) {
    // code to be executed if condition1 is true
    if (condition2) {
        // code to be executed if condition2 is true
    }
    else {
        // code to be executed if condition2 is false
    }
}
else {
    // code to be executed if condition1 is false
}
```



Prof. M. Iqbal Bhat (JKHED)



# if-else Statement: Variations

```
// else-if ladder
if (condition1) {
    // code to be executed if condition1 is true
}
else if (condition2) {
    // code to be executed if condition2 is true
}
else if (condition3) {
    // code to be executed if condition3 is true
}
.
.
.
else {
    // code to be executed if none of the above conditions are true
}
```

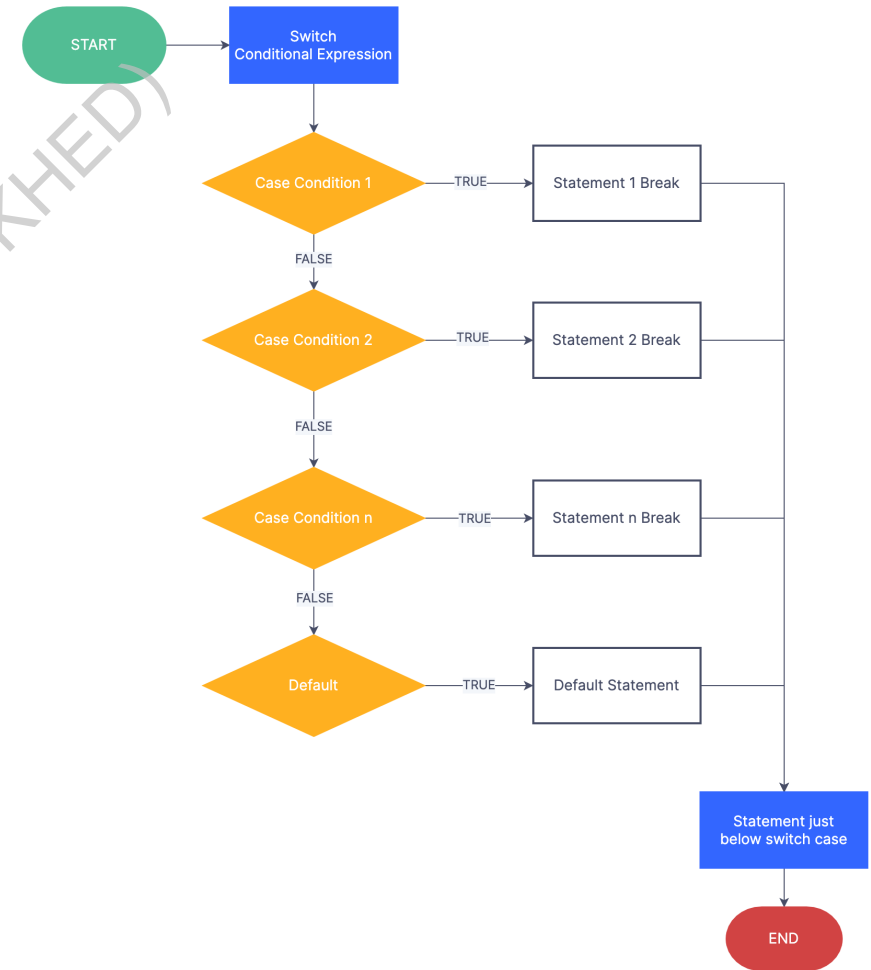


Prof. M. Lalal Bhat (JKHED)

# Switch Statement:

- The switch statement is used to execute a block of code based on the value of a variable or an expression..

```
switch (expression) {  
  case value1:  
    // value 1 code block  
    break;  
  case value2:  
    // value 2 code block  
    break;  
  .  
  .  
  .  
  default:  
    // code to be executed  
    //if none of the above cases  
    //are true  
}
```



+

o

# switch Statement:

Print the name of a day based on the value of a variable.

```
#include <stdio.h>
int main() {
    int day;
    printf("Enter a number between 1 and 7: ");
    scanf("%d", &day);
    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:
            printf("Wednesday\n");
            break;
        case 4:
            printf("Thursday\n");
            break;
        case 5:
            printf("Friday\n");
            break;
        case 6:
            printf("Saturday\n");
            break;
        case 7:
            printf("Sunday\n");
            break;
        default:
            printf("Invalid input.\n");
    }
    return 0;
}
```

+

●

○

Prof. M. Iqbal Bhat (JKUHP)

# Loops

Prof. M. Iqbal Bhat (JKHED)

# Loop Statements

- Loop statements are used to execute a block of code repeatedly until a certain condition is met.
  1. For loop
  2. While loop
  3. Do-while loop

Prof. M. Iqbal Bhat (JKUED)

# Loop Statements:

Loop statements are used to execute a block of code repeatedly until a certain condition is met

for-loop

while  
loop

do-while  
loop

Prof. M. Iqbal Bhat (JKHED)

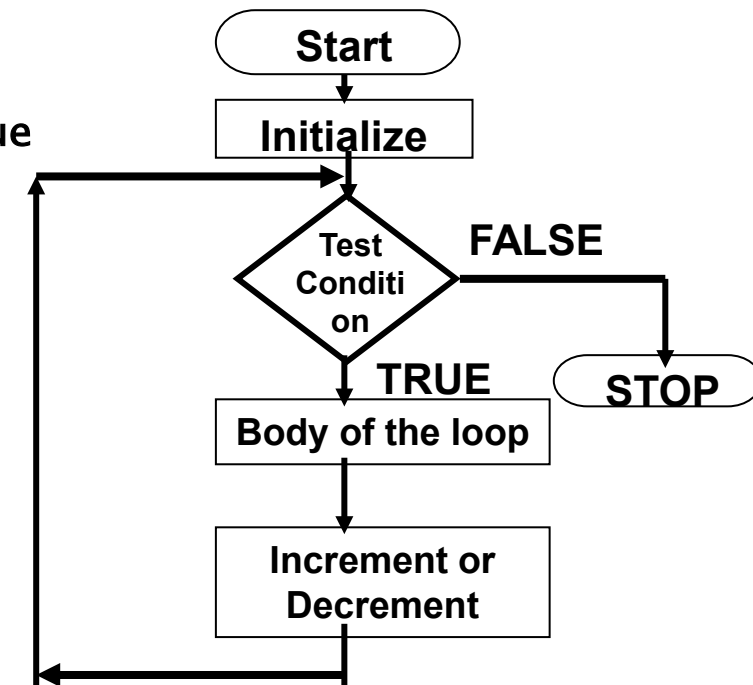
# for-loop

- The for loop is used to execute a block of code a specific number of times.
- It consists of three parts: the initialization statement (executed only once at the beginning of the loop), the condition statement (checked before each iteration), and the update statement (executed at the end of each iteration).

```
for (initialization; condition; update) {  
    // code to be executed repeatedly while the condition is true  
}
```

- Example:

```
for (int i = 0; i < 5; i++) {  
    printf("%d ", i);  
}  
// Output: 0 1 2 3 4
```



# for-loop: Variations

```
// For loop with multiple initialization statements
for (int i = 0, j = 0; i < 5; i++, j+=2) {
    printf("%d %d ", i, j);
}
```

```
// Output: 0 0 1 2 2 4 3 6 4 8
```

```
//For loop with empty initialization and update statements:
int i = 0;
for (; i < 5;) {
    printf("%d ", i);
    i++;
}
```

```
// Output: 0 1 2 3 4
```

Prof. M. Iqbal Bhat (JKHED)



# for-loop: Variations

```
// For loop with empty initialization, condition, and update  
statements (infinite loop):
```

```
for (;;) {  
    printf("This is an infinite loop\n");  
}
```

```
//For loop with nested loops
```

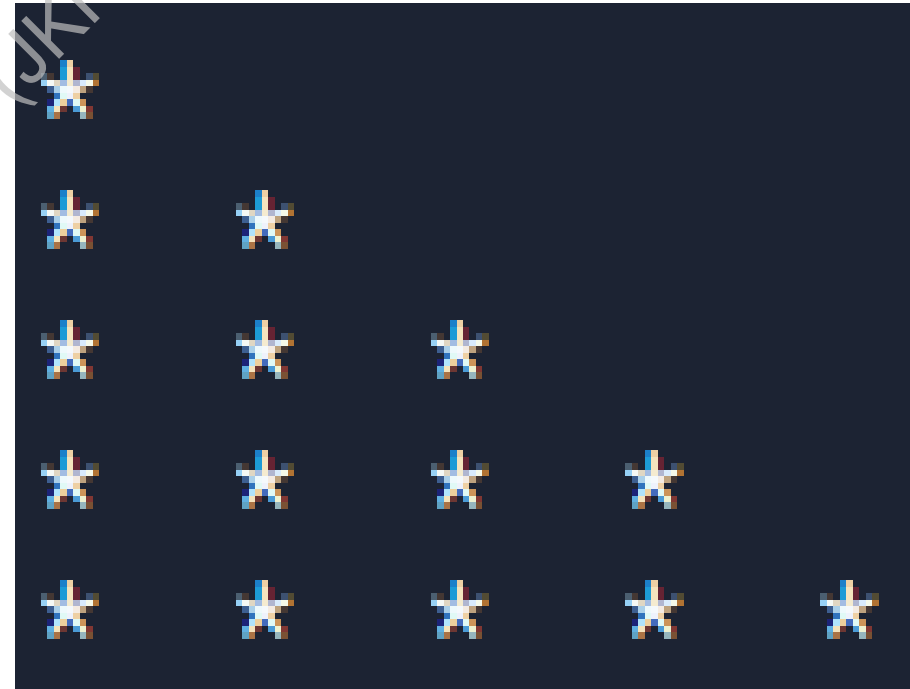
```
for (int i = 0; i < 5; i++) {  
    for (int j = 0; j < 3; j++) {  
        printf("%d %d ", i, j);  
    }  
}
```

```
// Output: 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2 3 0 3 1 3 2 4 0 4  
1 4 2
```

# Examples:

- Print Pascal Triangle (variation-1)

```
#include <stdio.h>
int main(void) {
    int i,j;
    printf("\n");
    for(i=1;i<=5;i++){
        for(j=1;j<=i;j++){
            printf("%-3c",'*');
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

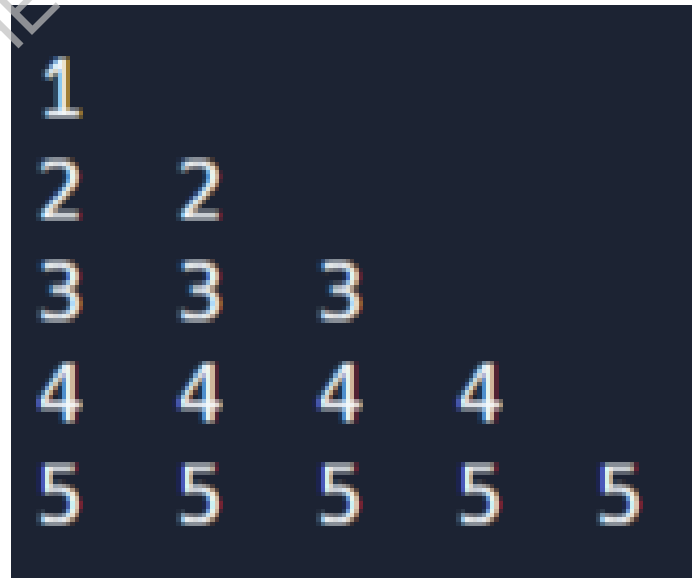


Prof. M. Iqbal Bhat (JKHED)

# Examples:

- Print Pascal Triangle (variation-2)

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



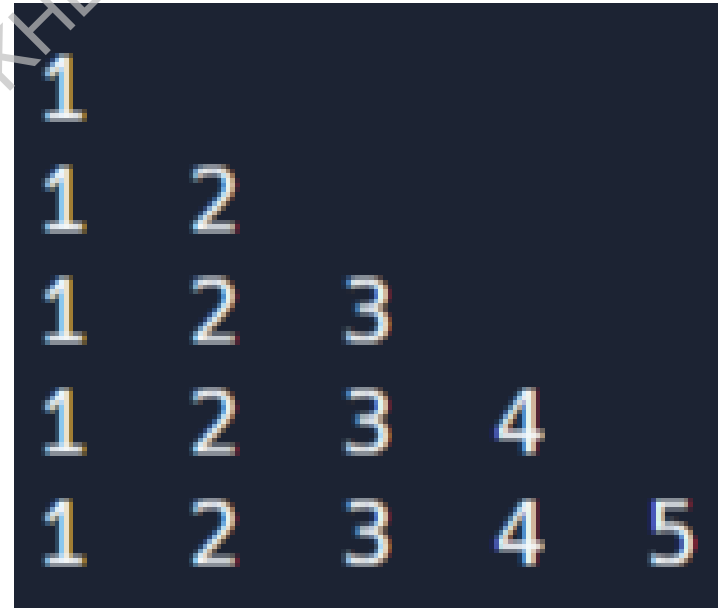
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Prof. M. Iqbal Bhat (JKHED)

# Examples:

- Print Pascal Triangle (variation-3)

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



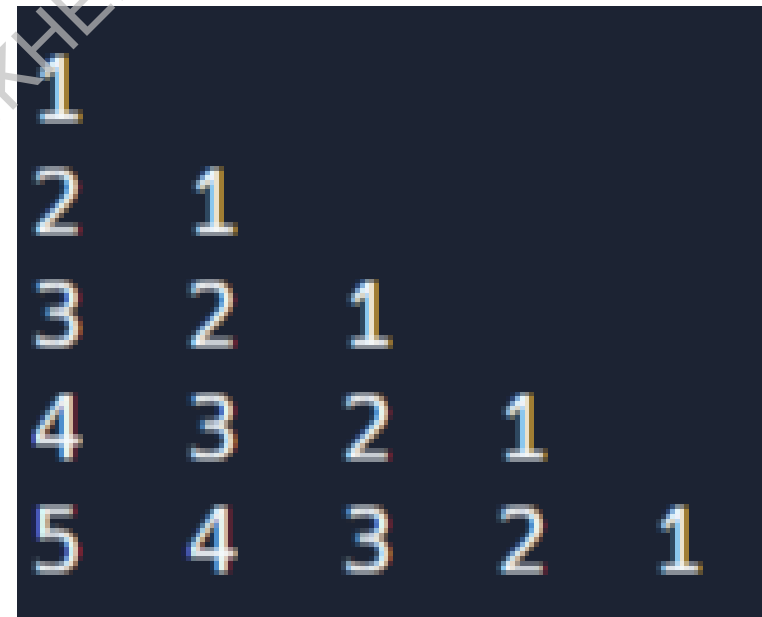
1  
1 2  
1 2 3  
1 2 3 4  
1 2 3 4 5

Prof. M. Iqbal Bhat (JKHED)

# Examples:

- Print Pascal Triangle (variation-4)

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



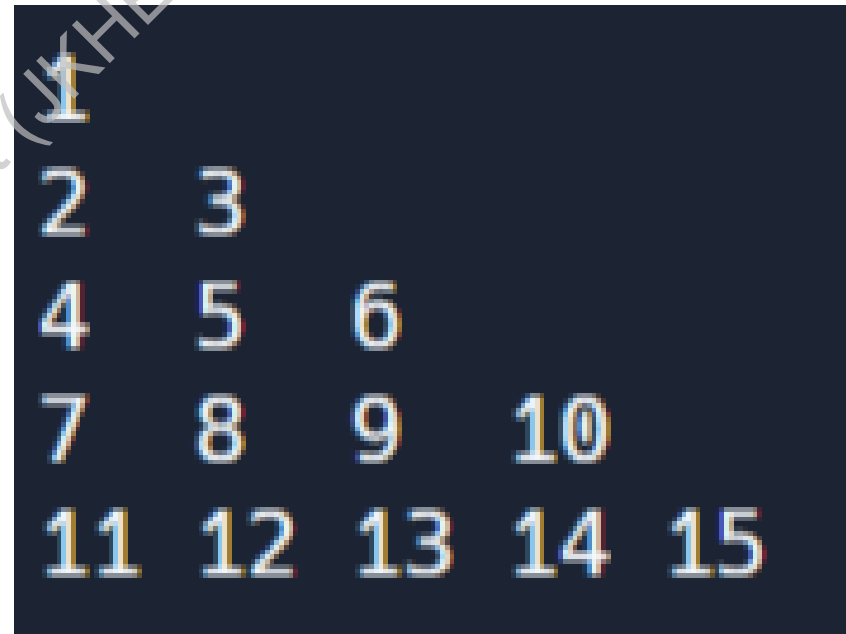
The output of the program is a 5x5 grid of numbers. Each row contains a sequence of numbers starting from the row number and decreasing by 1 until it reaches 1. The numbers are left-aligned in each row.

1				
2	1			
3	2	1		
4	3	2	1	
5	4	3	2	1

# Examples:

- Print Pascal Triangle (variation-5)

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



The output of the program is a 5x5 grid of numbers, where each row contains a sequence of numbers starting from 1 and increasing by 1 up to the row number. The numbers are displayed in a monospace font on a dark background.

1				
2	3			
4	5	6		
7	8	9	10	
11	12	13	14	15

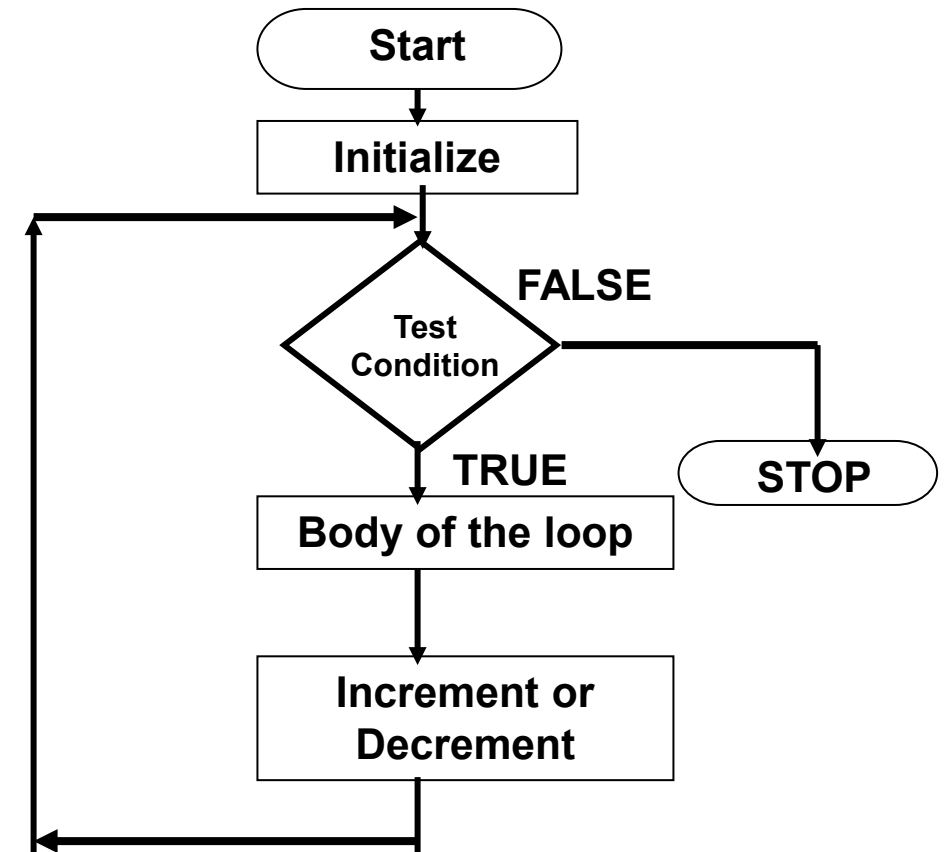
# while-loop

- While loop is used to execute a block of code as long as a certain condition is true.

```
while (condition)
{
    // code to be executed
}
```

Example:

```
int i=1;
while (i<=5) {
    printf("%d ", i);
}
// output: 1 2 3 4 5
```



# while-loop: Examples

```
#include <stdio.h>
```

```
int main() {
```

```
    int n, count = 0;
```

```
    printf("Enter a number: ");
```

```
    scanf("%d", &n);
```

```
    while (n != 0) {
```

```
        n /= 10;
```

```
        count++;
```

```
    }
```

```
    printf("The number of digits in the entered number is %d.", count);
```

```
    return 0;
```

```
}
```

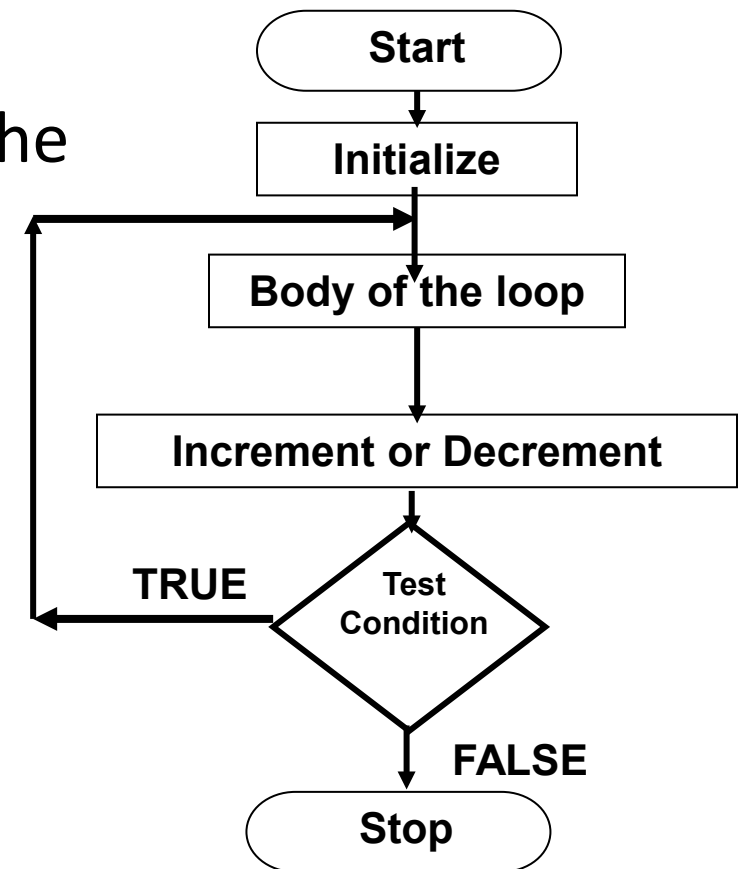
Prof. M. Iqbal Bhat (JKHED)



# do-while-loop

- In a do-while loop, the code inside the loop is executed at least once, and then the condition is checked. If the condition is true, the code inside the loop is executed repeatedly until the condition becomes false.

```
do {  
    // code to be executed repeatedly  
} while (condition);
```



# do-while-loop: Examples

```
#include <stdio.h>

int main() {
    int n, sum = 0, count = 0;
    float average;
    char choice;

    do {
        printf("Enter a number: ");
        scanf("%d", &n);

        sum += n;
        count++;

        printf("Do you want to enter another number? (Y/N): ");
        scanf(" %c", &choice);
    } while (choice == 'Y' || choice == 'y');

    average = (float)sum / count;
    printf("The average of the entered numbers is %.2f.", average);

    return 0;
}
```

# break; statement

- The **break** statement is used to terminate the execution of a loop or switch statement.
- It can be used inside a for loop, while loop, or do-while loop.
- When the **break** statement is encountered, the program control exits the loop and continues with the next statement after the loop.

```
#include <stdio.h>
int main() {
    int i;

    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            break;
        }
        printf("%d ", i);
    }
    return 0;
}
```

Prof. M. Iqbal Bhat (JAFED)

# continue; statement

- The continue statement is used to skip the current iteration of a loop and continue with the next iteration.
- It can be used inside a for loop, while loop, or do-while loop.
- When the continue statement is encountered, the program control skips the remaining statements in the loop body for the current iteration and continues with the next iteration..

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

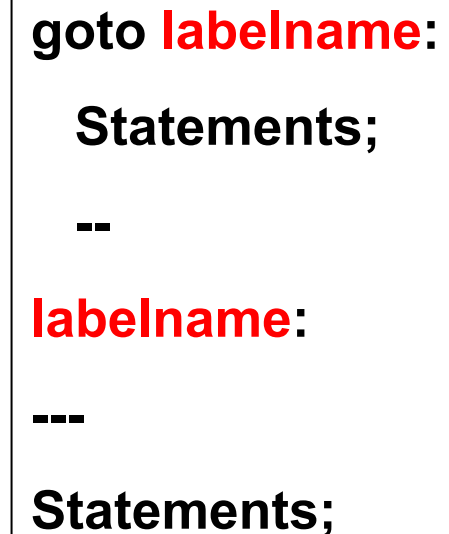
Prof. M. Iqbal Bhat (JKUHD)

# goto; statement

- The goto statement is used to transfer control to a labeled statement in the same function.
- It can be used to jump out of a nested loop or switch statement.
- However, it can make the code difficult to read and understand, and it can lead to spaghetti code.

```
goto label;  
...  
label: statement;
```

```
goto labelname:  
  Statements;  
  --  
  labelname:  
  ---  
  Statements;
```

A diagram illustrating the goto statement. It shows a code block with a 'goto labelname:' statement followed by 'Statements;' and a separator '--'. Below this is a label 'labelname:' followed by '---' and 'Statements;'. A curved arrow on the right side of the code block points from the 'labelname:' in the first line to the 'labelname:' in the second line, indicating the jump.

# goto statement: Examples

```
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i <= 10; i++) {
        if (i == 5) {
            goto end;
        }
        printf("%d ", i);
    }

    end:
    printf("Jumped to end label.");

    return 0;
}
```

Prof. M. Iqbal Bhat (JKHED)

# Assignment:

- **Write a “C” Program to generate Armstrong No.s from 1 to 1000.**

{Armstrong numbers, also known as narcissistic numbers, are a type of number in which the sum of the cubes of its digits is equal to the number itself}

- **Write a program to print a given number in reverse format.**

{1234 = 4321}

- **Factorial: This program calculates the factorial of a given number.**

# EXERCISES

Write a program to print the following Outputs using for while and do..while loops?

```
1
1 1
1 1 1
1 1 1 1
1 1 1 1 1
```

```
5
4 5
3 4 5
2 3 4 5
1 2 3 4 5
```

```
1
1 1
1 2 1
1 3 3 1
```

```
1
2 3 2
3 4 5 4 3
4 5 6 7 6 5 4
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
1
2 5 2
3 6 9 6 3
4 7 10 13 10 7 4
```



# Questions?

