

# LAB 2

## Experiment No. 1

### TITLE:

To demonstrate various decision control statements.

### OBJECTIVE:

- To learn different types of control statements in c.
- To learn to use them in program..

### THEORY:

Decision control statements allow the program to execute different sets of instructions based on certain conditions. There are mainly three types of decision control statements in C programming:

1. **if statement:** It executes a block of code if a specified condition is true.
2. **if-else statement:** It executes one block of code if the condition is true and another if it is false.
3. **switch statement:** It allows the program to select one of many blocks of code to be executed.

### PROGRAM:

#### Source Codes

- If statement

```
#include <stdio.h>

int main() {
    int number = 10;
    if(number > 0) {
        printf("The number is positive.\n");
    }
    return 0;
}
```

#### Output:

A screenshot of a terminal window with a black background. The text "The number is positive." is displayed in a light blue or cyan monospaced font. The word "is" is highlighted in a slightly different shade of blue.

The number is positive.

- If-Else Statement

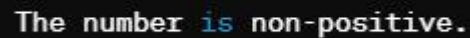
```
#include <stdio.h>

int main() {
    int number = -5;

    if (number > 0) {
        printf("The number is positive.\n");
    } else {
        printf("The number is non-positive.\n");
    }

    return 0;
}
```

Output:

A black rectangular box containing the text "The number is non-positive." in a monospaced font. The word "is" is highlighted in blue.

- Switch Statement

```
#include <stdio.h>

int main() {
    char grade = 'B';

    switch (grade) {
        case 'A': printf("Excellent!\n"); break;
        case 'B': printf("Well done!\n"); break;
        case 'C': printf("You passed.\n"); break;
        default: printf("Invalid grade.\n");
    }

    return 0;
}
```

Output:

A black rectangular box containing the text "Well done!" in a monospaced font. The word "done" is highlighted in blue.

## **RESULTS AND DISCUSSION:**

Decision control statements are fundamental in programming as they allow the program to make decisions based on certain conditions. The if statement is used for simple binary decisions, while the if-else statement allows for branching between two different outcomes. The switch statement is useful when there are multiple possible conditions to check against.

## **CONCLUSION:**

Through this lab, we have successfully demonstrated the implementation of various decision control statements in C programming. Understanding these statements is crucial for writing efficient and effective programs.

## Experiment No. 2

### TITLE:

To demonstrate various looping statement.

### OBJECTIVE:

- To learn concept of data types in C programming language.
- To learn how to declare, initialize, and manipulate different data types in C.

### THEORY:

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.

In C programming, there are 3 types of loop.

1. while loop
2. do-while loop
3. for loop

#### while loop

The while loop in C programming is used to execute a block of code repeatedly until a specified condition becomes false. The syntax of the while loop is as follows:

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

#### do-while loop

The do-while loop in C programming is similar to the while loop, except that the condition is evaluated after executing the loop body. This guarantees that the loop body executes at least once. The syntax of the do-while loop is as follows:

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

#### for loop

The for loop in C programming is used to iterate a part of the program several times. It consists of three parts: initialization, condition, and increment/decrement. The syntax of the for loop is as follows:

```
for (initialization; condition; increment/decrement) {  
    // code to be executed  
}
```

## PROGRAM:

### Source Codes

#### 1. While loop

```
#include <stdio.h>

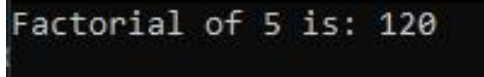
int main() {
    int num = 5;

    // Calculating factorial of a number using a while loop
    int fact = 1;
    int i = 1;
    while (i <= num) {
        fact *= i;
        i++;
    }

    printf("Factorial of %d is: %d\n", num, fact);

    return 0;
}
```

### Output:

A screenshot of a terminal window showing the output of the program. The text "Factorial of 5 is: 120" is displayed in a monospaced font on a dark background.

```
Factorial of 5 is: 120
```

#### 2. Do-while loop

```
#include <stdio.h>

int main() {
    int i = 0;

    // Iterating through an array using a do-while loop.
    printf("numbers printing using a do-while loop:\n");

    do {
        printf("%d ", i);
        i++;
    } while (i < 5);
    printf("\n");

    return 0;
}
```

### Output:

```
numbers printing using a do-while loop:  
0 1 2 3 4
```

### 3. For loop

```
#include <stdio.h>  
  
int main() {  
    int i;  
  
    // Printing numbers from 1 to 5 using a for loop  
    printf("Using for loop to print numbers from 1 to 5:\n");  
    for(i = 1; i <= 5; i++) {  
        printf("%d ", i);  
    }  
    printf("\n");  
  
    return 0;  
}
```

### Output:

```
Using for loop to print numbers from 1 to 5:  
1 2 3 4 5
```

## **RESULTS AND DISCUSSION:**

- The for loop is used when the number of replication is known beforehand.
- The while loop is suitable when the condition for termination is known before the loop starts.
- The do-while loop is similar to the while loop but guarantees that the loop body executes at least once before the condition is checked.

## **CONCLUSION:**

This lab successfully demonstrated the usage of various looping statements in C programming. Each loop serves different purposes and can be applied depending on the specific requirements of a program.

## Experiment No. 3

### TITLE:

To demonstrate goto, break and continue statements.

### OBJECTIVE:

- To learn how the goto, break and continue statements are work in C program.

### THEORY:

The C goto statement is a jump statement which is sometimes also referred to as an unconditional jump statement. The goto statement can be used to jump from anywhere to anywhere within a block of program

The break in C is a loop control statement that breaks out of the loop when encountered. It can be used inside loops or switch statements to bring the control out of the block. The break statement can only break out of a single loop at a time.

The continue statement in C is a jump statement that is used to bring the program control to the start of the loop. We can use the continue statement in the while loop, for loop, or do.. while loop to alter the normal flow of the program execution. Unlike break, it cannot be used with a C switch case.

### PROGRAM:

#### Source Codes

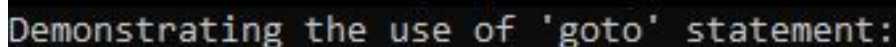
#### 1. Goto statement

```
#include <stdio.h>

int main() {
    goto jmp;
    printf("This message did not display.");
    jmp:
    printf("Demonstrating the use of 'goto' statement:\n");

    return 0;
}
```

#### Output:

A screenshot of a terminal window showing the output of the C program. The text displayed is "Demonstrating the use of 'goto' statement:" in a monospaced font on a black background.

```
Demonstrating the use of 'goto' statement:
```

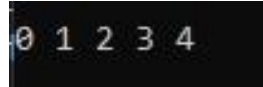
## 2. Break Statement

```
#include <stdio.h>

int main() {
    int i;

    // Demonstrating the use of 'break' statement
    for (i = 0; i < 10; i++) {
        if (i == 5)
            break;
        printf("%d ", i);
    }
}
```

### Output

A terminal window with a black background and white text showing the output of the program: 0 1 2 3 4. The cursor is positioned at the end of the output.

## 3. Continue Statement

```
#include <stdio.h>

int main() {
    int i;
    // Demonstrating the use of 'continue' statement
    for (i = 0; i < 10; i++) {
        if (i == 5)
            continue;
        printf("%d ", i);
    }

    return 0;
}
```

### Output

A terminal window with a black background and white text showing the output of the program: 0 1 2 3 4 6 7 8 9. The cursor is positioned at the end of the output.



## **RESULTS AND DISCUSSION:**

The program output demonstrates the behavior of goto, break, and continue statements within loops.

- The goto statement is used to jump from one point to another within a block.
- The break statement terminates the loop prematurely when a specific condition is met.
- The continue statement skips the current iteration of the loop and proceeds to the next iteration.

## **CONCLUSION:**

The experiment successfully demonstrated the use of goto, break, and continue statements in C programming language. These statements provide flexibility in controlling program flow and optimizing loop execution. However, their misuse can lead to code readability and maintenance issues, hence, it's essential to use them judiciously.