CL1002 *Programming Fundamentals Lab*	Lab 04 Basic Decision Structures

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Fall 2024

Author: Muhammad Nouman Hanif

AIMS AND OBJECTIVES

Aim:

The aim of learning basic decision structures in C is to understand and implement logical decision-making in programs. This involves using conditions to control the flow of execution, allowing programs to make choices based on different criteria, enhancing their interactivity, functionality, and responsiveness.

Objectives:

1. **Understand the Concept of Decision Structures:**
    - Grasp the importance of decision-making in programming and how it helps in controlling the flow of a program.
2. **Learn Basic Decision-Making Statements:**
    - Familiarize yourself with the primary decision structures in C, such as if, if-else, and switch statements.
3. **Implement if and if-else Statements:**
    - Learn how to use if statements to execute code blocks based on a condition.
    - Use if-else statements to handle multiple branches, allowing different actions based on whether a condition is true or false.
4. **Use switch Statements for Multiple Choices:**
    - Learn how to use switch statements for scenarios involving multiple possible cases, improving code readability and organization over multiple if-else conditions.
5. **Understand Relational and Logical Operators:**
    - Gain proficiency in using relational (==, !=, <, >, <=, >=) and logical operators (&&, ||, !) to form complex conditions within decision structures.

INTRODUCTION

Decision structures are fundamental building blocks in programming that allow programs to make choices based on certain conditions. In C, decision structures enable you to control the flow of execution, making your programs dynamic and responsive to different inputs and scenarios.

At the core of decision-making in C are statements like if, if-else, and switch, which provide mechanisms to execute specific blocks of code depending on whether certain conditions are true or false. These structures allow the program to "decide" what actions to take, making them essential for building interactive applications, performing calculations based on user input, managing error handling, and much more.

Understanding and effectively using decision structures is a critical skill for any programmer. They form the basis for writing logical, efficient, and flexible code that can adapt to varying requirements. Whether you are developing simple applications or complex systems, mastering decision structures will enhance your ability to create programs that can intelligently respond to different situations.

CONDITIONAL STATEMENTS

In C programming there are decision making statements, we need these kinds of statements because while programming we often need to make a lot of decisions. Let's take an example of a traffic signal management as show below

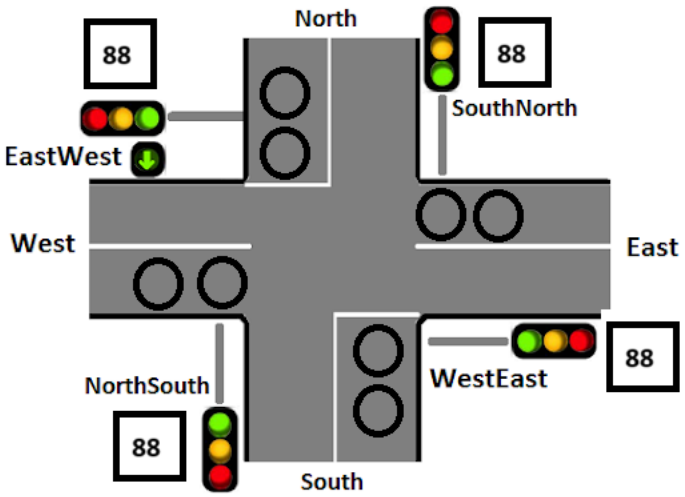


Figure 1. Example of a 4-Way smart traffic signal

In most countries there are vehicle sensors embedded in the road that detect how many cars are present in that lane. In the above-mentioned example these sensors are represented via black circles. If both circles are occupied by cars in that lane the time would be reduced to allow a smoother flow of traffic else if only one of the sensors is covered in that lane, then the timer would engage normally and start counting downwards until the counter value hits zero. Otherwise, if no sensors detect a vehicle, then the signal would remain red.

EXAMPLE 1

In some cases, we need to execute this block of code

```
01  If the number is odd
02  {
03      //Execute some code if both sensors are detecting a vehicle
04  }
05  else
06  {
07      //Execute some code
08  }
```

Otherwise, we want to execute this block

```
01  If the number is odd
02  {
03      //Execute some code
04  }
05  else
06  {
07      //Keep the signal light red if sensors detect no vehicle
08  }
```

In C, an if/else statement specifies that one block of code should be executed if a condition is true, and another block should be executed if that condition is false.

To write meaningful if/else statements, it is important to know operators which allow us to compare two expressions and produce a Boolean outcome.

In C, however, there are no distinct values for true or false, instead a false condition is denoted by a number 0 and anything which is non-zero is considered true.

expr1 == expr 2	This condition checks if expr1 is equal to expr2
expr1 != expr 2	This condition checks if expr1 is not equal to expr2
expr1 < expr 2	This condition checks if expr1 is less than expr2
expr1 <= expr 2	This condition checks if expr1 is less than or equal to expr2
expr1 > expr 2	This condition checks if expr1 is greater than expr2
expr1 >= expr 2	This condition checks if expr1 is greater than or equal to expr2
!expr1	This condition checks the logical NOT of expr1
expr1 && expr 2	This condition checks the logical AND of expr1 and expr2
expr1 || expr 2	This condition checks the logical OR of expr1 and expr2

Table 1. Use cases for if-else if- else statement

In 'C' programming conditional statements are possible with the help of the following two constructs:

1. If statement

2. If-else statement

It is also called branching as a program decides which statement to execute based on the result of the evaluated condition.

IF-STATEMENT

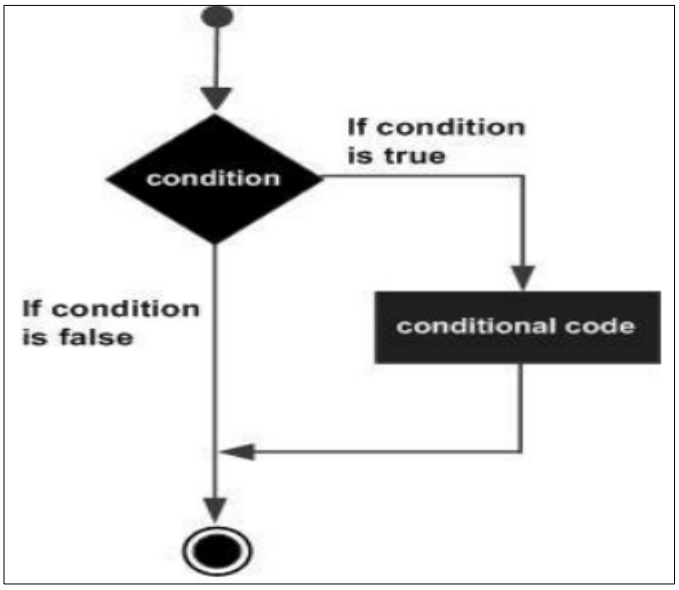
An if statement consists of a conditional expression followed by one or more statements.

If the conditional expression evaluates to true, then the block of code inside the if statement will be executed. If the conditional expression evaluates to false, then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Syntax: The syntax of an if statement in C programming language is:

```
If (condition)
        {
                //statements;
         }
```

Flowchart:



EXAMPLE 2

Checking if the number input by user is 0 or not. If it's 0 then print Zero else print non-zero

```
01 #include <stdio.h>
02 int main() {
03     int num1, num2;
04     printf ("Enter two integers \n");
05     scanf ("%d%d",&num1,&num2);
06     if(num2!=0){
07         printf("num1/num2 = %d\n",num1/num2);
08     }
09     return 0;
10 }
```

IF-ELSE STATEMENT

An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

Syntax: The syntax of an if...else statement in C programming language is:

```
If (condition)
{
//statements;
}
else
{
//statements;
}
```

EXAMPLE 3

Check if one number is greater than the other.

```
01 #include <stdio.h>
02 int main() {
03    int num1, num2;
04    printf ("Enter two integers \n");
05    scanf ("%d%d",&num1,&num2);
06    if(num2>num1){
07       printf("num2 = %d is greater\n",num2);
08    }else{
09       printf("num1 = %d is greater ",num1);
10    }
11    return 0;
12 }
```

ELSE IF STATEMENT

An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single **if...else if** statement.
When using **if, else if, else** statements there are few points to keep in mind:

- An **if** can have zero or one else's and it must come after any **else ifs**.
- An **if** can have zero to many **else if's** and they must come before the **else**.
- Once an **else if** succeeds, none of the remaining **else if's** or **else's** will be tested.

EXAMPLE 4

Syntax: The syntax of an if...else statement in C programming language is:

```
09 #include <stdio.h>
10 int main() {
11    int num1;
12    printf ("Enter value =\n");
13    scanf ("%d",&num1);
14    if(num1>0){
15       printf("num1 = %d is positive\n",num1);
16    }else if (num1<0){
17       printf("num1 = %d is negative ",num1);
18    }else {
19       printf("num1 = %d is zero ",num1);
20    }
21
22    return 0;
23 }
```

SWITCH-CASE-STATEMENTS

Another way that programs can make decisions is to use switch/case. The syntax of switch/case is shown in the figure below.

```
switch (selection expression) {
    case 1:
        //statement
        break;
    case 2:
        //statement
        break;
    default:
        //statement
            }
```

Here, when the execution arrow reaches the switch statement, the selection expression—in parenthesis after the keyword switch—is evaluated to a value.

This value is then used to determine which case to enter. The execution arrow then jumps to the corresponding case—the one whose label (the constant immediately after the keyword case) matches the selection expression's value. If no label matches, then the execution arrow jumps to the default case if there is one, and to the closing curly brace of the switch if not.

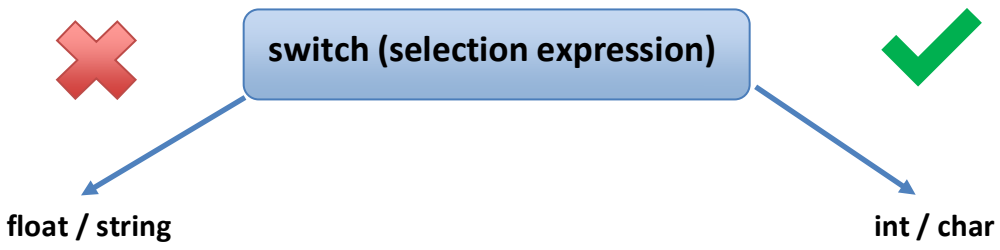


Figure 2: Switch case data type requirement

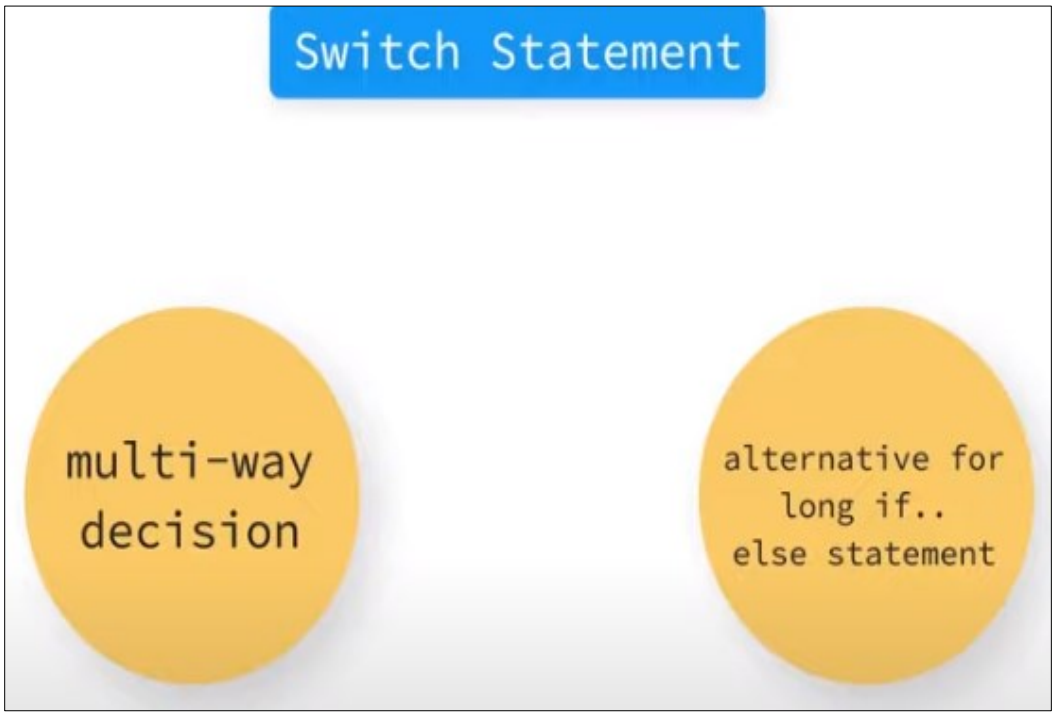
The following rules apply to a **switch** statement:

- The **expression** used in a **switch** statement must have an integral or enumerated type or be of a class type in which the class has a single conversion function to an integral or enumerated type.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.

Switch statement is better than if else statement

A switch statement is **usually more efficient than a set of nested ifs**

Switch statement acts as a substitute for a long **if-**else-if ladder that is used to test a list of cases.



EXAMPLE 5

```c
#include <stdio.h>
int main() {
    int day;  // Variable to store the user's input
    printf("Enter a number (1-7) to get the corresponding day of the week: ");
    scanf("%d", &day);
    // Switch case to determine the day of the week
    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! Please enter a number between 1 and 7.\n");
    }
    return 0;
}
```

Note on every case (except default) we have to use the break keyword as the program will keep checking for other cases even if the first case is verified.

LAB Exercise

1. Write a C program to check whether a number is multiple of 3 or not. If it is then print “This number is multiple of 3”, otherwise print “This number is not multiple of 3”.

2. Create a calculator asking for operator (+ or – or * or /) and operands and performs calculation according to the user input using switch statement.

3. Write a C program to input a character from the user and check whether the given character is a small alphabet, capital alphabet, digit, or special character, using if else.

4. An online shopping store is providing discounts on the items due to the Eid. If the cost of items is less than 2000 it will give a discount up to 5%. If the cost of shopping is 2000 to 4000, a 10% discount will be applied. If the cost of shopping is 4000 to 6000, a 20% discount will be applied. If it's more than 6000 then a 35% discount will be applied to the cost of shopping. Print the actual amount, saved amount and the amount after discount. The Minimum amount eligible for a discount is 500.

5. Write a program in C to calculate and print the Electricity bill of a given customer. The customer id., name and unit consumed by the user should be taken from the keyboard and display the total amount to pay to the customer. The charges are as follow:

Unit	Charge/Unit
Up to 199	@16.20
200 and above but less than 300	@20.10
300 and above but less than 500	@27.10
500 and above	@35.90

If the bill exceeds Rs. 18000 then a surcharge of 15% will be charged on top of the bill.

Test Input:

```
1001 //Customer ID
James //Customer Name
800 //Units Consumed
```

***Expected Output*:**

```
Customer ID :1001
Customer Name: James
Units Consumed :800
Amount Charges @Rs. 35.90 per
unit: 28720
Surcharge Amount: 4308
Net Amount Paid by the
Customer: 33028.00
```

6. Given a positive integer denoting n, do the following:
    a. If 1<=n<=9, print lowercase English words corresponding to the numbers e.g. (one for 1, two for 2)
    b. B. If n>9 print greater then 9

7. An android developer wants to design a mobile feature to control the brightness of the mobile phone according to the surrounding light. In order to do it he uses an ambient light sensor (for the detection of surrounding light) which is commonly built in in all major android phones. It gives the value of light intensity in integers. Write a C program for Light sensor value ranges from 0-1000, if it's exposed under sunshine (>500), if it's evening then (0 ~ 100), lighting (100 to 500).

8. Write a program to see greetings according to time using a 24-hour format. If the time between is 5 to 11 it should greet "Good Morning", if time is between 12 to 18 it should greet "Good Evening", if time between 18 to 24 it should greet "Good Night".

9. Write a program in which user enters his NTS and F.Sc marks and your program will help student in selection of university. Based on these marks Student will be allocated a seat at different department of different university.
    a. Oxford University:
        i. IT: Above 70% in Fsc. And 70 % in NTS
        ii. Electronics: Above 70% in Fsc. And 60 % in NTS

iii. Telecomunication: Above 70% in Fsc. And 50 % in NTS

b. MIT:
   i. IT: 70% - 60 % in Fsc. And 50 % in NTS
   ii. Chemical: 59% – 50 % in Fsc. And 50 % in NTS
   iii. Computer: Above 40% and below 50 % in Fsc. And 50 % in NTS

10. Write a C program that takes the temperature as input from the user and prints a message based on the temperature range:
    a. Temperature < 0: “Freezing weather”
    b. 0 to 10: “Very cold weather”
    c. 1 to 20: “Cold weather”
    d. 1 to 30: “Normal temperature”
    e. 1 to 40: “Hot weather”
    f. 40: “Very hot weather”