1. Introduction to Flow Control

Flow control refers to the **order in which statements of a program are executed**. By default, a C program executes statements **sequentially (line by line)**. However, real-life problems require decision-making and branching.

Flow control statements allow a program to:

- Make decisions
- Execute different blocks of code
- Control the program execution path

---

2. Need for Decision Making in C

Decision making is required when:

- Different actions are needed for different conditions
- Programs must respond to user input
- Logical conditions must be tested

Example (Real Life)

- If it is raining → take an umbrella
- Else → do not take an umbrella

In C language, this is achieved using **if–else statements**.

---

3. Types of Flow Control Statements in C

C language supports three types of flow control statements:

1. **Decision Control Statements**
    - if
    - if–else
    - nested if
    - else–if ladder
    - switch
2. **Looping Statements**
    - for
    - while
    - do–while

3. **Jump Statements**
    - break
    - continue
    - goto
    - return

This chapter focuses on **if–else decision control statements**.

---

4. Simple if Statement

The **if statement** is used to execute a block of code **only when a condition is true**.

Syntax

```
if(condition)
{
   statements;
}
```

Working

- Condition is evaluated
- If condition is **true (non-zero)** → statements execute
- If condition is **false (zero)** → statements are skipped

Example

```
int a = 10;
if(a > 5)
{
   printf("a is greater than 5");
}
```

---

5. if–else Statement

The **if–else statement** allows execution of **one block when condition is true** and **another block when condition is false**.

Syntax

```
if(condition)
{
   statements1;
}
else
{
   statements2;
}
```

Flowchart Explanation

1. Condition is checked
2. If true → if block executes
3. If false → else block executes

Example

```
int num = 7;
if(num % 2 == 0)
{
   printf("Even number");
}
else
{
   printf("Odd number");
}
```

---

6. Nested if–else Statement

When an **if or else block contains another if–else**, it is called **nested if–else**.

Syntax

```
if(condition1)
{
   if(condition2)
   {
      statements;
   }
   else
   {
      statements;
   }
}
else
{
   statements;
}
```

Example

```
int a = 10, b = 20;
if(a > b)
{
   printf("a is greater");
}
else
{
   if(b > a)
   {
      printf("b is greater");
   }
   else
   {
      printf("Both are equal");
   }
}
```

7. else–if Ladder

The **else–if ladder** is used when **multiple conditions** need to be checked.

Syntax

```
if(condition1)
{
   statements;
}
else if(condition2)
{
   statements;
}
else if(condition3)
{
   statements;
}
else
{
   statements;
}
```

Working

- Conditions are checked **from top to bottom**
- First true condition block executes
- Remaining conditions are skipped

Example

```
int marks = 85;
if(marks >= 90)
   printf("Grade A");
else if(marks >= 75)
   printf("Grade B");
else if(marks >= 60)
   printf("Grade C");
else
   printf("Fail");
```

---

8. if–else vs else–if Ladder

Feature	if–else	else–if ladder
Conditions	Single	Multiple
Usage	Simple decisions	Multiple decisions
Complexity	Low	Moderate

9. Conditional Expressions in if Statement

Conditions in if statements use:

- Relational operators (<, >, ==, etc.)
- Logical operators (&&, ||, !)

Example

```
if(age > 18 && age < 60)
{
   printf("Eligible");
}
```

10. Use of Braces in if–else

- Braces {} are optional for single statement
- Recommended for clarity and safety

Example

```
if(x > 0)
   printf("Positive");
else
   printf("Negative");
```

11. Common Programming Examples

11.1 Check Positive or Negative

```
if(num > 0)
   printf("Positive");
else
   printf("Negative");
```

11.2 Largest of Two Numbers

```
if(a > b)
   printf("a is largest");
else
   printf("b is largest");
```

11.3 Leap Year Check

```
if(year % 4 == 0)
   printf("Leap Year");
else
   printf("Not Leap Year");
```

12. Common Errors in if–else

1. Using = instead of ==
2. Missing braces
3. Wrong logical conditions
4. Incorrect nesting
5. Semicolon after if condition

Example of Error

```
if(a = 5)   // Wrong
```

13. Advantages of if–else Statement

- Easy to understand
- Improves program logic
- Useful for decision making
- Reduces code complexity

14. Limitations of if–else Statement

- Becomes complex for many conditions
- Hard to manage deeply nested if–else
- switch statement may be better in some cases

15. Conclusion

Flow control using **if–else statements** is a fundamental concept in C programming. It allows programs to make decisions and execute appropriate code blocks based on conditions. A strong understanding of if–else statements helps in writing efficient, logical, and error-free programs.