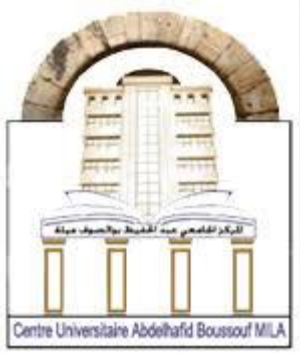
University
Center of Mila



Structure of Computers and Applications
1st year ST – LMD & ENG

Part 2: The basics of Algorithm and Program

Course 10: CONTROL STRUCTURES / STATEMENTS

I. Decision Making Statements

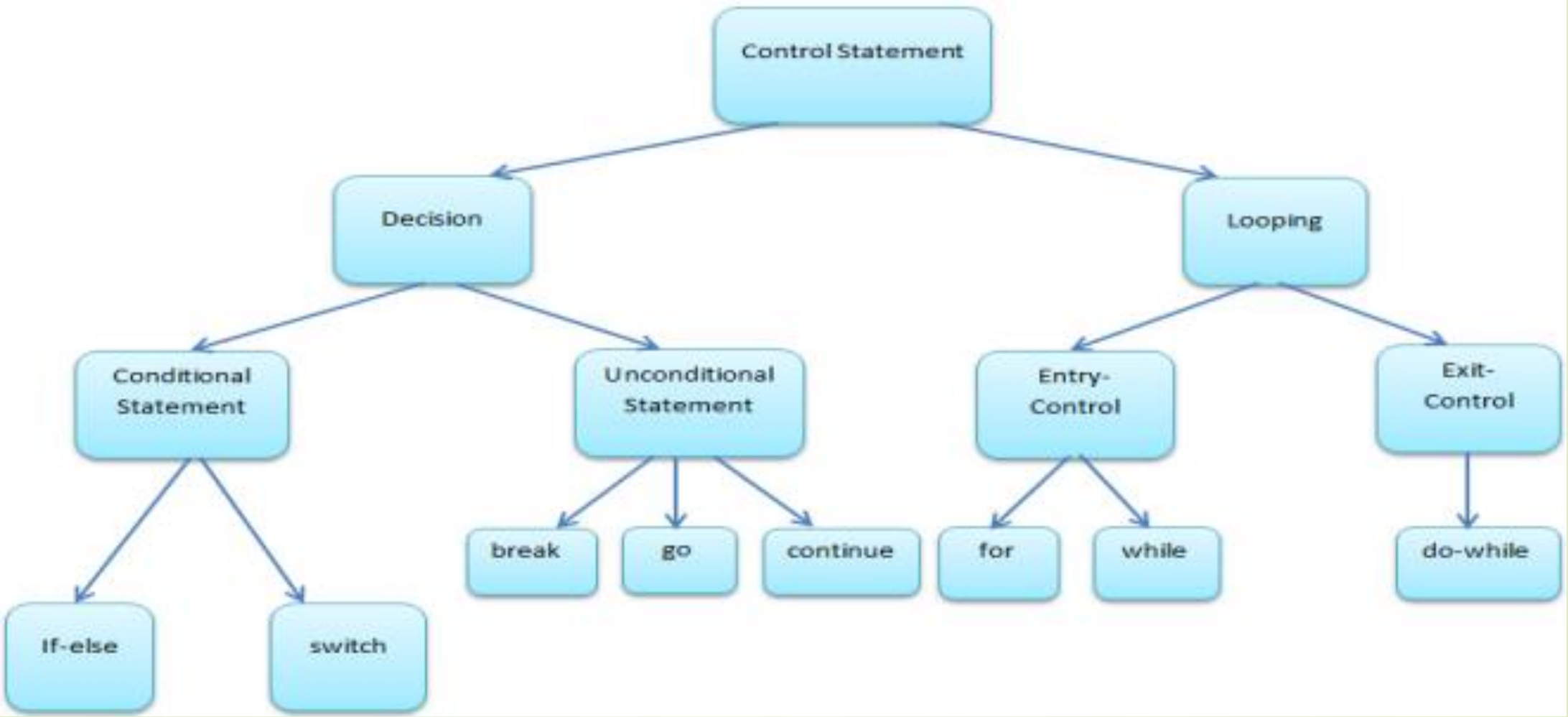
By

Dr. Farouk KECITA

Academic year : 2025/2026

Introduction

- These are the statements which alter the normal (sequential) execution flow of a program.



I. Decision Making Statements

- These statements allows us to make decision based upon result of a condition
- These statements can be called decision making or conditional statements or sometimes called as selection statements, because based on condition it selects the set of instructions to be executed.
- C language supports two conditional statements : **if** & **switch**

i. **if Statement:** The if Statement may be implemented in different forms:

1. simple **if** statement.
2. **if – else** statement
3. **nested if – else** statement.
4. **else if ladder**.

i. if Statement

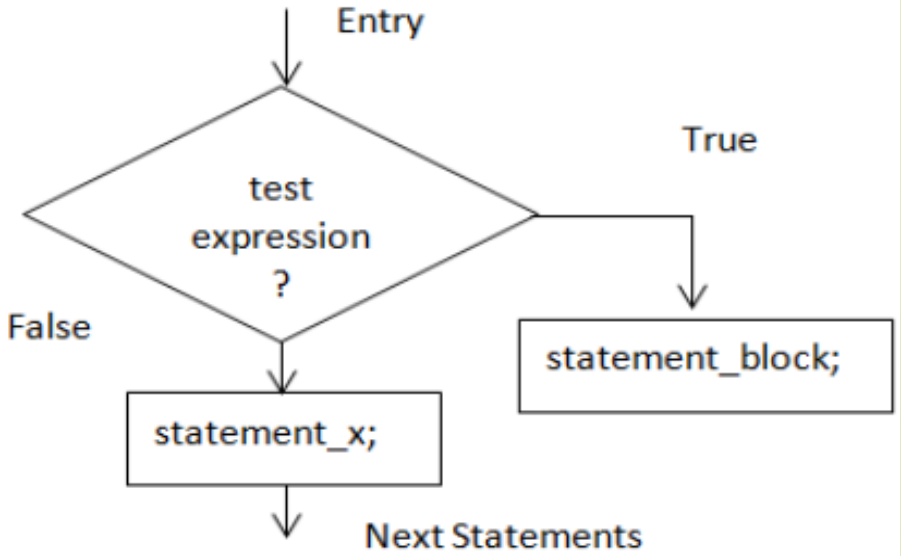
1. Simple if statement

- The if statement controls conditional branching
- It is used to execute the code **if condition is true**
- If the expression / condition is evaluated to **false** (0), statements inside the body of if is **skipped** from execution. And the execution will **jumped to** the **remaining** statements of the program.

Syntax:

```
if(condition)
        {
         Statement-1;
         ..............
         ..............
         Statement-n;
        }
   statements;
```

Flowchart for Simple If:



i. if Statement

1. Simple if statement

Example: write a c program to check whether student is passed or failed.

```c
/*Program to check whether student
is passed or failed*/
 #include <stdio.h>
 int main() {
 int marks;
 printf("Enter student marks: ");
scanf("%d",&marks);
if(marks>50) {
  printf("Student Passed \n");
}
if(marks<50){
printf("Student Failed \n");
}
  return 0; }
```

Output for **Run 1**

```
Enter student marks: 60
Student Passed
```

Output for **Run 2**

```
Enter student marks: 30
Student Failed
```

i. if Statement

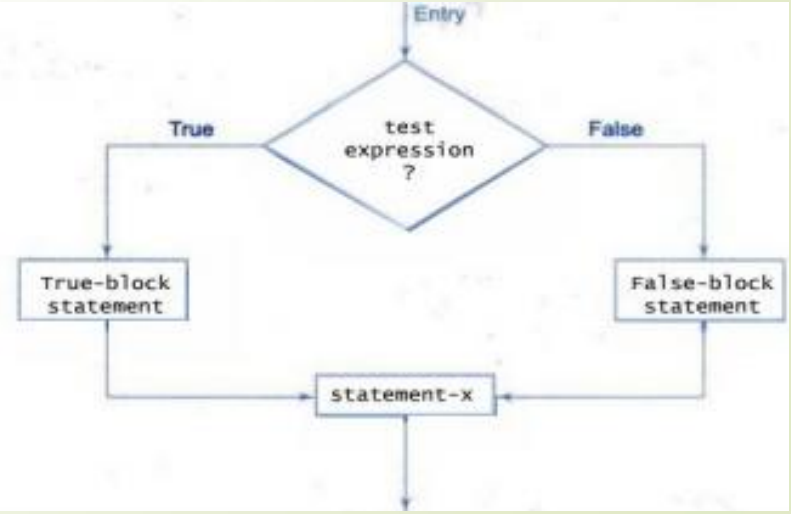
2. if – else statement

- The if-else statement is an **extension** of the 'simple if' statement.
- If the **test_expression** is **true**, then the **true-block**-statement(s), immediately following the if statement are **executed**; otherwise the **false-block**-statement(s) are executed.
- In **either case**, either true-block-statements or false-block-statements will be executed, **not both**.

Syntax:

```
if(condition)
      {
       true block statments;
      }
 else{
        false block statements;
      }
   statements;
```

Flowchart:



i. if Statement

2. if – else statement

Example: Write a C Program to check whether given number is even or odd

```c
/*Program to check whether given
 number is even or odd*/
#include<stdio.h>
 int main() {
int num;
printf("Enter number: ");
scanf("%d",&num);
if(num %2 == 0){
printf("%d is even number \n",num);
}
else {
printf("%d is odd number \n",num);
}
return 0; }
```

Output for **Run 1**

```
Enter number: 3
3 is odd number
```

Output for **Run 2**

```
Enter number: 10
10 is even number
```

i. if Statement

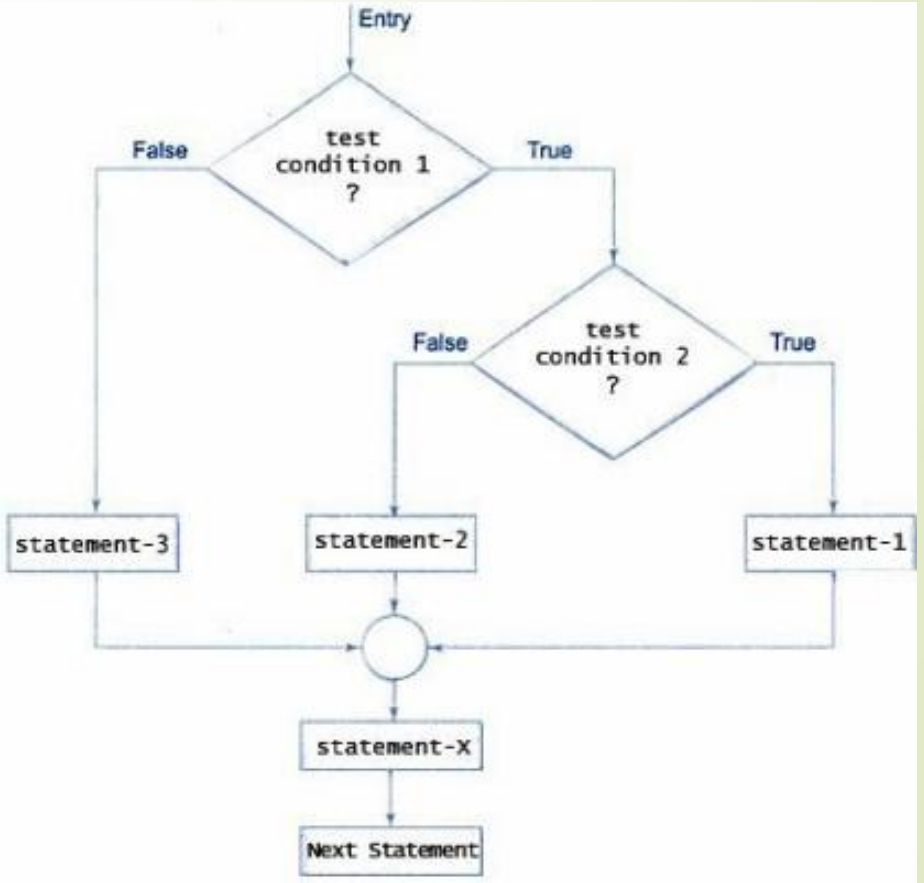
3. Nested if – else statement

- When a series of decisions are involved, we may have to use more than one if…else statement in nested form as follows:

Syntax:

```
if(condition1)
      {
  if(condition2)
       {
       if(condition3)
         {
            Statements1;
          }
       else{
            statements2;
            }
         }
       }
  statements;
```

Flowchart:



i. if Statement

3. Nested if – else statement

- If condition1 is **true** then condition2 **tested**, if condition2 is **true** then condition3 is **tested**, if condition3 also **true** then statements1 will be **executed** otherwise the statements2 will be **executed** i.e. when condition3 is **failed**.
- If condition1 is **false** then it **does not executes** any of the statements **inside it**, it directly **goes to out** of condition and **executes** the statements **out of it**.

Example:

```
#include<stdio.h>
int main() {
 int i = 10;
   if (i == 10) {
       if (i < 15)
           printf("i is smaller than 15\n");
       if (i < 12)
           printf("i is smaller than 12 too\n");
       else
           printf("i is greater than 15");
   }
   else {
       if (i == 20) {
           if (i < 22)
               printf("i is smaller than 22 too\n");
           else
               printf("i is greater than 22");
       }
   }
return 0; }
```

Output for **Run 1**

```
i is smaller than 15
i is smaller than 12 too
```

i. if Statement

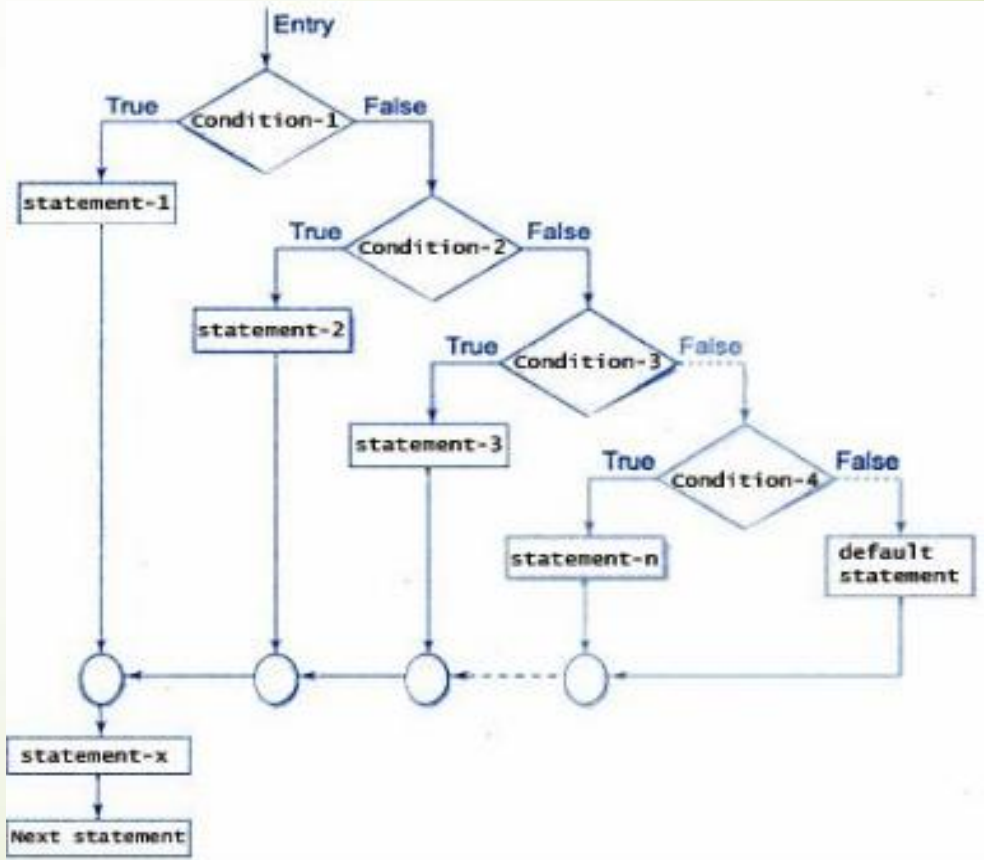
4. else if ladder

- There is another way of putting if's **together** when multipath decisions are involved.
- A multipath decision is a **chain** of if's in which the statement associated with each **else** is an **if**.

Syntax:

```
if(condition1)
        {
         statements1;
        }
else if(condition2)
        {
         statements2;
        }
else if(condition3)
        {
         statements3;
        }
else    {
         default statements;
        }
```

Flowchart:



i. if Statement

4. else if ladder

- This construct is known as else if ladder.
- The conditions are evaluated from the top downwards.
- As soon as a true condition is found, the statement associated with it is executed and the control is transferred to statement-x.
- When all the n conditions become false, then the final else containing the default statement will be executed.

i. if Statement

4. else if ladder

Example: Write a C Program to illustrate concept of else-if ladder to select color

```c
#include<stdio.h>
int main() {
  int n;
printf("Enter a num b/w 1 & 4 to select color");
scanf("%d",&n);
if(n==1) {
printf("You selected Red color \n");
}
else if(n==2) {
printf("You selected Green color \n");
}
else if(n==3) {
printf("You selected yellow color \n");
}
else if(n==4) {
printf("You selected Blue color \n");
}
else {
printf("No color selected \n");
}
return 0; }
```

Output for **Run 1**

```
Enter a num b/w 1 & 4 to select color
3
You selected yellow color
```

Output for **Run 2**

```
Enter a num b/w 1 & 4 to select color
6
No color selected
```

I. Decision Making Statements

ii. Switch Statement

- The switch statement is a multiway branch statement.
- In the program there is a possibility to make a choice from number of options, then this structured statement is useful.

Syntax:

```
switch(expression)
     {
       case value1: statements1;
                    break;
        case value2: statements2;
                    break;
       …
       …

       case valuen: statementsn;
                    break;
           default: default statements; // optional
      }
statements;
```

ii. Switch Statement

- The expression in the switch is an **integer** expression or **characters** expression.
- Value1, value2, …., **value n** are **constants** or **characters** known as **case labels**.
- Note that case label are **ends** with a colon(**:**).
- When the switch is **evaluated**, the value of expression is **compared** against the value1, value2,…,value n. If a value **matches** with any **case**, then the block of statements are **executed**.
- The **break statement** at end of each block **signal** the **end** of particular case and causes an **exit** from the **switch** statement and transfers **control** to the **remaining** statements of the program.
- The **default** case is **executed** when **value** of expression **does not match** with any of the **case values** in **switch** statement.

ii. Switch Statement

Example : Write a C Program to print words corresponding numbers below 9

```c
/*Print words corresponding numbers below 9*/
#include<stdio.h>
int main() {
int n;
printf("Enter a number(0-3): ");
scanf("%d",&n);
switch(n)
{
case 0: printf("Zero\n");
break;
case 1: printf("One\n");
break;
case 2: printf("Two\n");
break;
case 3: printf("Three\n");
break;
default: printf("More than 3 \n");
}
return 0; }
```

Output for **Run 1**

```
Enter a number(0-3): 2
Two
```

Output for **Run 2**

```
Enter a number(0-3): 5
More than 3
```