10

CHAPTER

DECISION MAKING-BRANCHING

In a program, all the instructions are executed sequentially by default, when no repetitions of calculations are necessary. This type of execution is known as sequential execution. In some situations, there may be a chance to change the execution order of statements based on condition or to repeat a set of statements until certain conditions are met. This led to a kind of decision making to see whether a particular condition has occurred or not and then change the order of execution accordingly. Such facility allows us to:

- ❖ Select a set of statements from several options.
- ❖ Skip certain statements depending on some conditions in the program and continue execution of program from some other point.
- ❖ Repeat a section of program for a known number of times or until a specified condition is satisfied.

‘C’ language possesses such decision-making capabilities by supporting the following statements:

- ❖ if statement
- ❖ switch statement
- ❖ conditional operator statement
- ❖ goto statement

These statements are popularly known as decision-making statements. Since these statements control the flow of execution, they are also known as control statements.

10.1 DECISION MAKING USING ‘IF’ STATEMENT

The **if**statement is a powerful decision making statement and is used to control the flow of execution of statements. It statement is implemented in different forms depending on the complexity of the conditions to be tested. The different forms are

1. simple **if** statement.
2. **if …. else** statement.

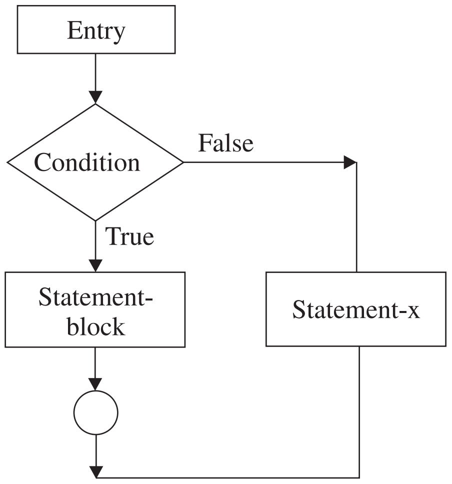
3. Nested **if ….else**statement is also called as **if** …. **else** ladder.
4. **else if** ladder.

10.1.1 Simple if statement

The **if** statement is a two-way decision making statement and is used in conjunction with an expression. It takes the following form:

Syntax

```
If(test expression)
{
statement-block;
}
statement-x;
```



Properties of an if statement

1. If the condition is true,then the statement inside the loop is executed.
2. If condition is false,the statement will be skipped and the execution will jump out of the loop.
3. If a compound structure is provided, it must be enclosed in opening and closing braces.

Example program for if statement

```
#include<stdio.h>
void main()
{
int i;
printf("enter the number less than 10..");
scanf("%d",&i);
if(i<=10)
{
printf("%d",i);
printf("\n the entered number %d is less than 10",i);
}
}
```

Output

```
Enter the number less than 10…5
The entered number 5 is less than 10.
```

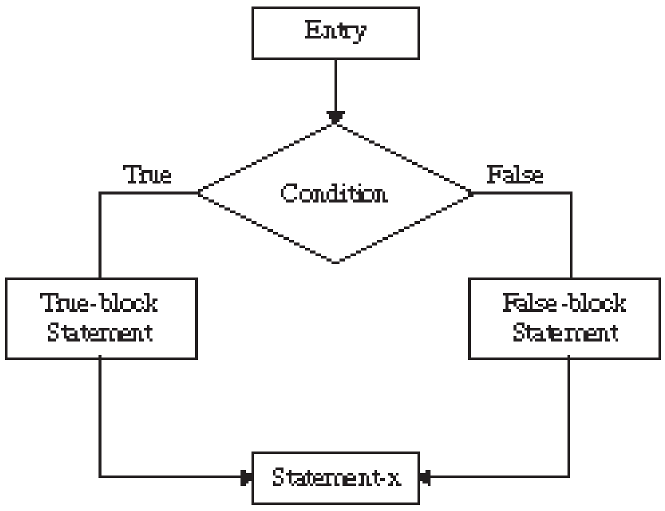
10.1.2 if-else statement

It is an extension of the simple **if** statement. The general form is:

Syntax

```
if(test expression)
{
true-block statement(s);
}
else
{
false-block statement(s);
}
statement-x;
```

If the test expression is true, then the true-block statement(s), immediately following the **if** statements, are executed; otherwise, the false-block statement(s) are executed. In either case, either true-block or false-block will be executed, not both. In both the cases, the control is transferred subsequently to the statement-x.



Example program for if…else (Given number is odd or even)

```
#include<stdio.h>
#include<conio.h>
void  main()
{
int num,rem;
clrscr();
printf("Enter the number");
scanf("%d",&num);
rem=num%2;
if(rem==0)
printf("The number is even");
else
printf("\n The number is odd");
getch();
}
```

Output

```
Enter the number: 80;
The number is even.
```

10.1.3 Nested if-else statement

In the **if-else** statement, if the body of either **if** or **else** or both includes another if-else statement, the statement is known as nested if.

They are used when there are two or more alternatives to select.

Syntax

```
If(condition 1)
{
if(condition 2)
    {
true statement;2
    }
else
    {
false statement2
    }
else
  {
false statement 1;
  }
}
```

Example program for nested If..Else

```
#include<stdio.h>
void main()
{
inta,b,c,max;
clrscr();
printf("\nEnter the values of a,b,c");
scanf("%d%d%d",&a,&b,&c);
if(a>b)
```

```
max=a;
else
max=b;
if(c>max)
max=c;
printf("The biggest number is %d",max);
getch();
```

Output

```
Enter the values of a,b,c
4
5
6
The biggest number is 6
```

10.1.4 The if-else-if ladder

Multiway decision based on several conditions is made using the **else if** variant of the **if** statement. This works by cascading several comparisons. As soon as one of these gives a true result, the following statement or block is executed, and no further comparisons are performed. If none of the conditions is true, the final **else**is executed. The final **else**statement is optional. If final **else** is not present, no action takes place if all other conditions are false.

Syntax

```
if (expression)
statement;
else if (expression)
statement;
else if (expression)
statement;
else if (expression)
statement;
else
statement;
```

Example program for if-else-if statement

```
#include<stdio.h>
int main(){
int numb1, numb2;
  printf("Enter two integers to check\n");
  scanf("%d %d",&numb1,&numb2);
if(numb1==numb2) //checking whether two integers are equal.
printf("Result: %d = %d",numb1,numb2);
else
if(numb1>numb2) //checking whether numb1 is greater than numb2.
  printf("Result: %d > %d",numb1,numb2);
  else
  printf("Result: %d > %d",numb2,numb1);
return 0;
}
```

Output

```
Enter two integers to check.
5
3
Result: 5 > 3
```

10.2 DECISION MAKING USING SWITCH CASE STATEMENT

It is a multiway decision statement, whether an expression matches one of a number of constant integer values, and branches accordingly. It is well structured but can only be used in certain cases where:

- ❖ Only one variable is tested, all branches must depend on the value of that variable. The variable must be an integral type(int,long,short or char).
- ❖ Each possible value of the variable can control a single branch. Default branch may optionally be used to trap all unspecified cases.

Syntax

```
switch(expression)
{
case 1:              integer or character variable
block1;          ──→ or an expression
break;------------- causes exit from switch
case 2:
block 2;
break;
.
.
.
default:
default block;
break;
}
```

The keyword **switch** is followed by a switch variable or expression in parenthesis. Each **case** keyword is followed by a constant, which is not in parenthesis but is followed by a colon. When the **switch** statement is executed, the expression is evaluated and its value is successively compared with the case constants. When a match is found, the statement sequence associated with that case is executed until the break statement or the end of the switch is reached. The **default** statement is executed if no matches found. The **default** statement is optional, and if it is not present, no action takes place if all matches fail. The **default** statement may appear anywhere within the switch and it is not necessarily to be placed at the end.

The **break** statements inside the **switch** statement are optional. They terminate the statement sequence associated with each constant. If the **break** statement is omitted, execution will continue into the next case statements until either a **break** or the end of the switch is reached.

Rules:

1. The expression in **switch** statement must be an integer value or character value.
2. No real number.
3. Each **case** block and default **block** must be terminated by break.
4. The default is optional.
5. The **case** keyword must end with (**:**).
6. Switch can be nested.
7. No two case constants are identical.

Example program for Switch Case Statement

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,option;
clrscr();
printf("\n Enter the values of a and b");
scanf("%d %d",&a,&b);
printf("\n 1.Addition");
printf("\n 2.Subtraction");
printf("\n 3.Multiplication");
printf("\n 4.Division");
printf("\n Enter your option");
scanf("%d",&option );
switch(option)
{
case 1:
c=a+b;
break;
case 2:
c=a-b;
break;
case 3:
c=a*b;
break;
case 4:
c=a/b;
}
Printf("\n The resultant value is %d",c);
getch();
}
```

Output

```
Enter the values of a and b
7
9
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your option3
The resultant value is 63
```

Example: Program to print the number up to 5 using break

```
#include<stdio.h>
void main()
{
int i;
for(i=1;i<=10;i++)
{
if(i==6)
break;
printf("%d",i);
}
}
```

Output

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

Important Points:

1. The **switch** differs from the **if,**in that switch can only test for equality, whereas **if** can evaluate any type of relational or logical expression.
2. No two **case** constants in the same **switch** can have identical values. However, a **switch** statement enclosed by an outer **switch**(i.e., nested switch) may have **case** constants that are in common.
3. If character constants are used in the **switch** statement, they are automatically converted to integers.
4. In **switch** statement, the **case** labels that contain **double** values or **strings** are not permitted.

10.3 DECISION MAKING USING CONDITIONAL OPERATOR

The conditional operator is used for two-way decisions. It is a combination of ? and : and takes three operands. The general form of the conditional operator is as follows:

conditionalexpression? expression1:expression2

The conditional expression is evaluated first. If the result is nonzero, expression1 is evaluated and is returned as the value of the conditional expression. Otherwise expression2 is evaluated and its value is returned.

For example, the segment,

```
if(x<0)
flag=0;
else
flag=1;
```

can be written as

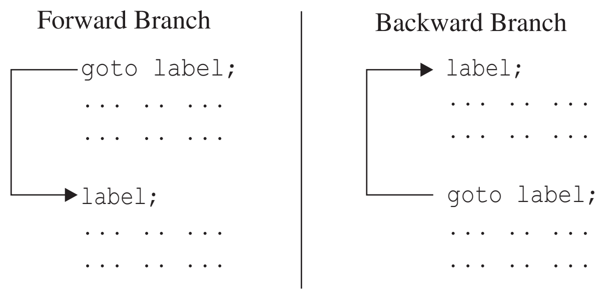
```
flag=(x<0)? 0:1;
```

10.4 THE GOTO STATEMENT

The **goto**statement is used to alter unconditionally the normal sequence of a program execution by transferring control to some other part of the program. Although it may not be essential to use the **goto**statement in a highly structured language like C, there may be occasions when the use of **goto**might be desirable.

Syntax

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



```
goto label;
... .. ...
... .. ...

label;
... .. ...
... .. ...
```

```
label;
... .. ...
... .. ...

goto label;
... .. ...
... .. ...
```

As shown above, the **goto**statement requires a *label* tospecify the place where the branch is to be made. A labelis a valid identifier followed by a colon. The label must be in the same function as the **goto**that uses it wherejump between functions is not possible.

The labelcan be anywhere in the program either before or after the **goto***label;* statement. A **goto**breaks the normal sequential execution of the program. If the label: is before the statement **goto***label;*a loop will be formed and some statements will be executed repeatedly.Such a jump is known as a *backward jump*. On the other hand, *i*f the *label:* is after the statement **goto***label;* some statements will be skipped and the jump is known as *forward jump.*

Example program for goto statement

```
int main()
{
int age;
Vote:
printf("you are eligible for voting");
No Vote:
printf("you are not eligible to vote");
printf("Enter you age:");
scanf("%d",&age);
if(age>=18)
goto Vote;
else
goto No Vote;
return 0;
}
```

A **goto**is often used at the end of the program to direct the control to go to the input statement to read further data. Another use of **goto**statement is to transfer the control out of a loop when certain peculiar conditions are encountered.

```
-----
while(……..)
{
for(…….)
{
---
----
if(----)gotoend_of_program;
--
}
----                              Jumping out of lops
-----
}
end_of_program;
```

gotostatement may be used to enhance the readability of the program to improve the execution speed.