UNIT II - Managing simple Input and Output operations - Operators and Expressions - Decision Making: Branching statements, looping statements - Function: Declaration, Definition - Passing arguments by value - Recursion - Storage classes.

2.3 DECISION MAKING STATEMENTS

The order in which the program statements are executed is known as flow of control. Decision making statements alters the default flow of control. There are 2 types of decision-making statements. They are,

- ✓ **Conditional Branching Statements**
- ✓ **Unconditional Branching Statements**

(I) Conditional Branching Statements

In conditional branching, program control is transferred from one point to another based upon the outcome of the condition. The conditional branching statements are:

a) if Statement

b) if-else Statement

c) Nested if Statement

d) if else if else statement

e) switch Statement

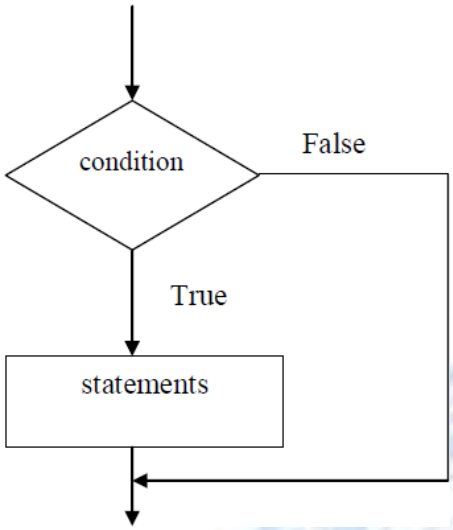
a) Simple if Statement

It check the given condition in if statement. If it is true then it will execute the body of if statement, otherwise it skipped the body of if statement.

Syntax:

```
if (condition)
{
Statement block
}
```

Flow Chart:



Program : Program To Check Whether the Number Is Positive

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int a;
printf (“\n Enter a number : ”);
scanf (“%d”, &a);
if ( a > 0)
printf (“The given number is positive number”);
getch();
}
```

Output

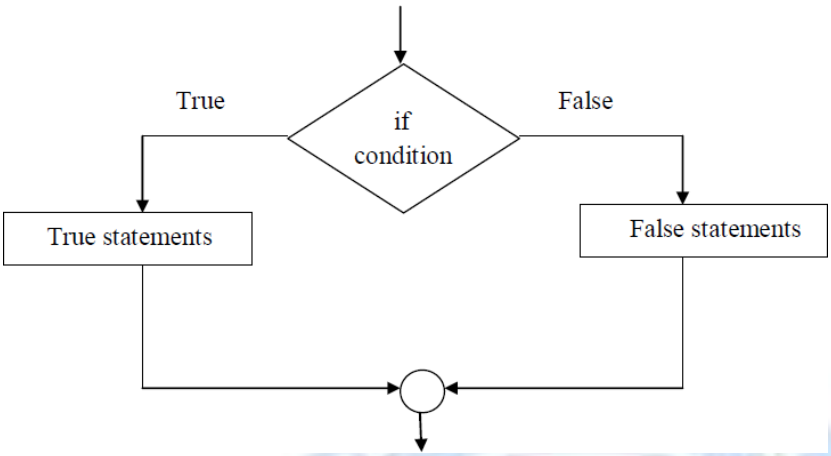
Enter a number :7

The given number is positive number

b) if - else Statement

It is a two way branching statement. If the condition is true then the True part statement will be executed.If the condition is false then the False part statement will be executed.

Flowchart:



Syntax:

```
if (condition)
{
True part Statement
}
else
{
False part Statement
}
```

Program: Check Whether the Number Is Odd or Even

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int n, r;
printf (“\n Enter a Number:”);
scanf (“%d”, &n);
r = n % 2;
if ( r == 0 )
printf (“Given Number is Even”);
else
printf (“Given Number is Odd”);
```

```
getch();
}
```

Output:

Enter a Number: 6

Given Number is Even

c) Nested if Statement

The if statement within another if statement is called as nested if statement.

Syntax:

```
if ( condition1 )
{
if ( condition2)
{
Inner if True part Statement
}
else
{
Inner if False part Statement
}
}
else
{
Outer if False part Statement
}
```

It checks the condition1 and if it is true it check the inner if condition2. This type of nested if is useful when a series of decisions are involved.

Program :

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int Mark;
printf (“Give your Mark”);
scanf (“%d”, &Mark);
```

```
if ( Mark < 50 )
printf("Failed");
else
{
if ( Mark < 60 )
printf("Second Class");
else
printf ("First Class");
}
getch();
}
```

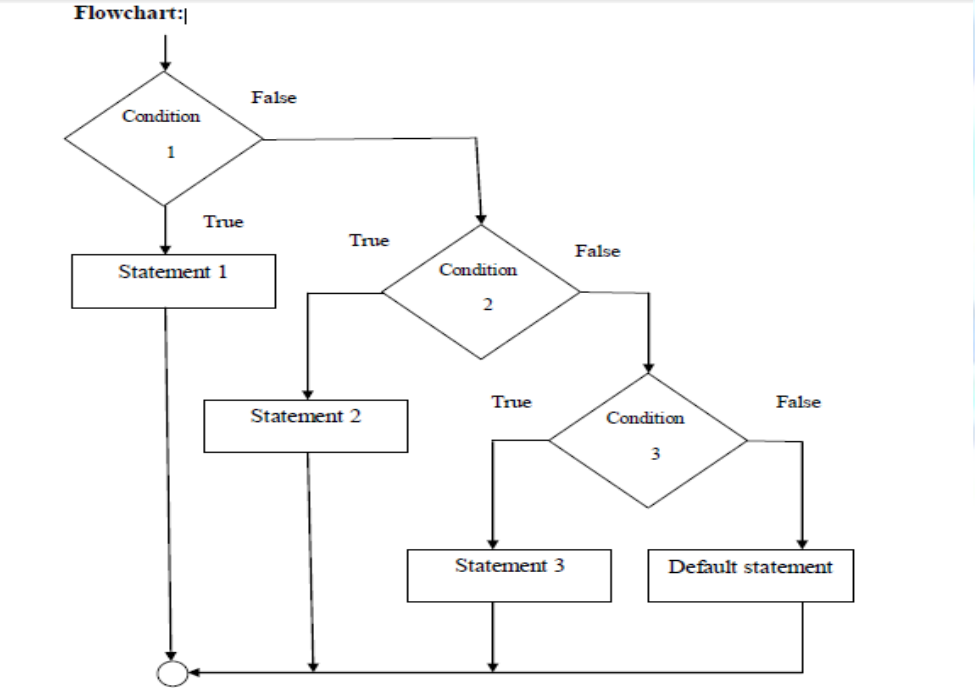
Output

Give your mark : 68

First class

d) if-else if- else statement

If the else part of if statement contain another if statement, then the else and the if statement can be combined. It is called else if ladder.



Syntax :

if (condition1)

```
Statement block 1
else if ( condition2)
Statement block 2
else if ( condition3)
Statement block 3
else
Statement block 4
```

If the condition1 evaluated is true, the statement block1 is executed. If the condition2 is true, then statement block2 is executed and so on. If none of the conditions are true, then the statement block4 is executed.

Program : Find Largest Among Three Numbers

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int a, b, c;
printf (“Enter three numbers : ”);
scanf (“%d %d %d”, &a, &b, &c);
if (a > b) && (a > c)
printf(“Biggest Number is %d”, a);
else if (b > c)
printf(“Biggest Number is %d”, b);
else
printf (“Biggest Number is %d”, c);
getch();
}
```

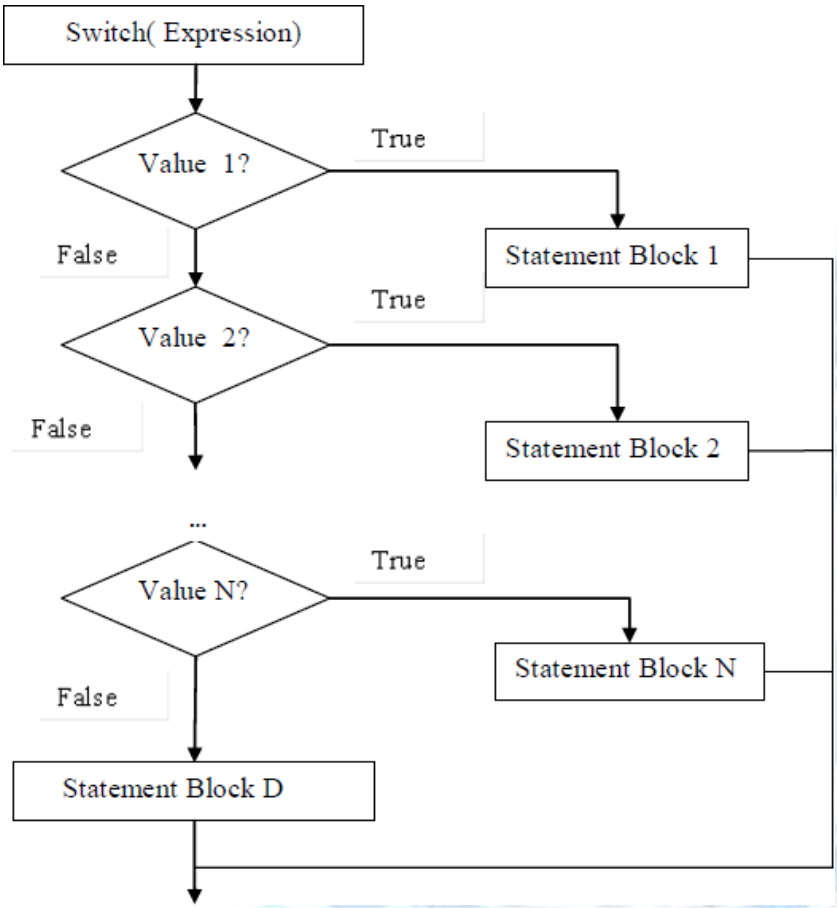
Output :

```
Enter three numbers : 40 -50 35
Biggest Number is 40
```

e) switch Statement

It is a multiway branching statement. It first evaluates the expression in switch statement.

That result is compared with each case value one by one. Whenever a match found, it execute the statements given in the corresponding case statement.If none of the case value matches with the result it executes the default section.



Syntax:

switch (Expression)

{

case value 1:

Statement block 1

break;

case value 2:

Statement block 2

break;

. . .

case value n:

Statement block n

break;

default:

Default Statement block }

Writing Switch Statement

 The expression used in switch statement must be an integer or a character data.

 The case labels must be character or integer constant.

 Each case block must be terminated by break statement. Otherwise, all statements that are followed by matched cases are executed.

 The default clause is optional & usually placed at the end.

 The case keyword must terminate with colon (:)

 No two case constants are identical.

Program : Perform Arithmetic Operation Using Switch statement

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
float value1, value2;
char operator;
printf (“Type your expression : ”);
scanf (“%f %c %f”, &value1, &operator, &value2);
switch(operator)
{
case „+“ :
printf (“%f”, value1 + value2);
break;case „-“ :
printf (“%f”, value1 - value2);
break;
case „*“ :
printf (“%f”, value1 * value2);
break;
```

```
case „/“ :
printf (“%f”, value1 / value2);
break;
default :
printf (“unknown operator”);
}
getch();
}
```

Output

Type your expression: 10 + 5

15

(II) Unconditional Branching Statement

In an unconditional branching, program control is transfer from one point to another without checking the condition. Following are the unconditional branching statements.

i) goto

ii) break

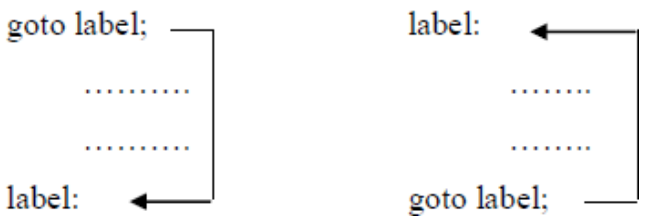
iii) continue

iv) return

i) goto Statement

- C“ provides the goto statement to transfer control unconditionally from one place to another place in the program.
- The goto statement can move the program control almost anywhere in the program.
- The goto statement requires a label.

Syntax:



Program : Check Whether the Given Number is Prime or Not Using goto & return.

```
# include<stdio.h>
```

```
# include<conio.h>
void main ( )
{
int No, i;
printf (“Give the number : ”);
scanf (“%d” , &No);
for ( i = 2 ; i <= No / 2; i++ )
{
if ( No / i == 0 )
goto stop;
}
printf (“ Given Number is a Prime Number”);
return;
stop : printf (“ Given Number is not a Prime Number”);
}
```

Output:

Give the number : 17

Given Number is a Prime Number

ii) break Statement

- It is used within a looping statement or switch statement.
- The break statement is used to ***terminate the loop***.
- In switch statement each case block must be terminated with break statement to exit from switch.

Syntax:

break;

Example:

Refer switch example program.

iii) continue Statement

- It is used within looping statements.

- When the continue statement is used inside the loop, it skip the statements which are available after this statement in the loop and go for the next iteration.

Syntax:

continue;

Program : Display 1 To 10 Except 5

```
# include<stdio.h>
#include<conio.h>
void main ( )
{
int i;
for (i =1; i <= 10; i++ )
{
if ( i == 5 )
continue;
printf (" %d ", i);
getch();
}
```

Output:

1 2 3 4 6 7 8 9 10

break	continue
Break statement takes the control to the outside of the loop	Continue statement takes the control to the beginning of the loop.
It is used in loop & switch statements	This can be used only in loop statements

return Statement

A return statement terminates the execution of a function and returns the control to the calling function.

The general form of a return statement is

return;

OR

return expression;

OR

return(expression);

2.1 LOOPING STATEMENTS

- The loop is defined as the block of statements which are repeatedly executed for a specified number of times or until a particular condition is satisfied.
- If there is a single statement in the loop, the blocking braces is not necessary. If more than one statement in the loop then the loop statements must be placed within braces.

The following are the loop statements in „C".

a) for

b) while

c) do – while

a) for Loop

We use for loop when we know exactly how many times the loop statements are repeated.

Syntax:

```
for (initialization; condition; incrementing / decrementing)
{
Statements
}
```

Initialization:

It has the initial value for the counter variable. In a for loop, initialization is executed first. It is executed only once i.e., for the first iteration only.

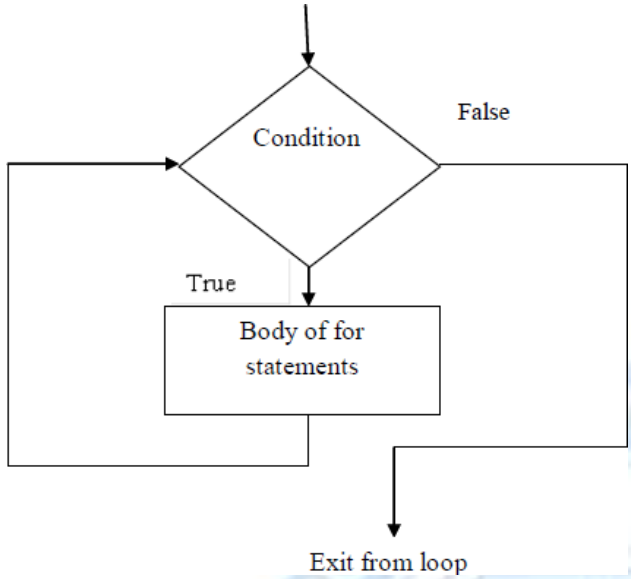
Condition:

The condition represents a test expression.

Incrementing / decrementing:

After completing every iteration, the counter variable must be increased or decreased. Otherwise it may leads to an infinite loop.

Flowchart



Program: Print the Sum of the Series 1 + 2 + 3 + 4 . . . up to N terms

```
#include<stdio.h>
#include<conio.h>
void main ( )
{
int i, sum = 0, n;
printf (“Enter the number of terms”);
scanf (“%d”, &n);
for (i = 1; i <= n; i++ )
sum = sum+ i;
printf (“ Sum = %d”, sum);
getch();
}
```

Output

Enter the number of terms 5

Sum=15

b) while Loop

while loop is a pre testing loop. The conditional expression is tested before the body is executed. If the condition is true the loop will be repeated otherwise stop the iteration. If the very first time itself the condition failed the loop will not be executed at least one time.

Syntax:

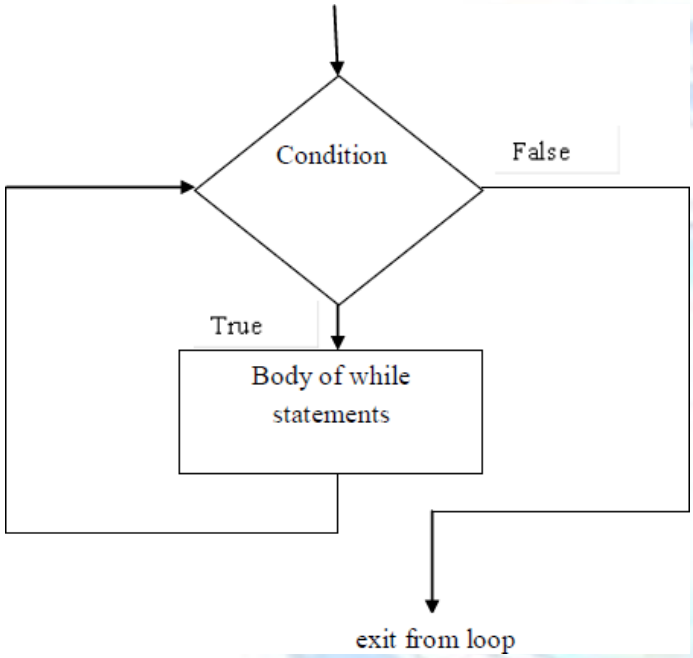
while (condition)

{

Body of the loop

}

Flowchart:



Program: Print the Sum of the Series 1 + 2 + 3 + 4 . . . up to N terms

```
#include<stdio.h>
#include<conio.h>

void main ( )
{
int i, sum = 0, n;
printf (“Enter the number of terms”);
scanf (“%d”, &n);
i=1;
while( i <= n)
{
sum = sum+ i;
i++;
```

```
}
printf (" Sum = %d", sum);
getch();
}
```

Output

Enter the number of terms 5

Sum=15

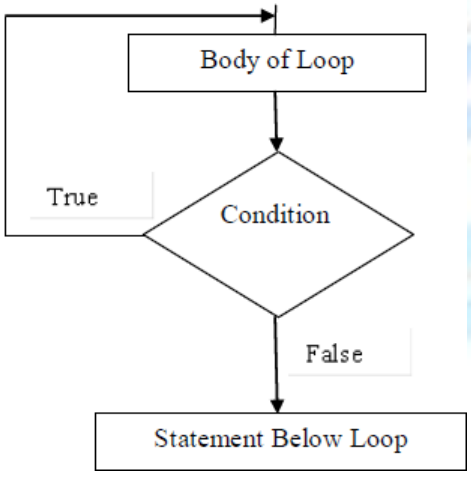
c) <u>do…while Loop</u>

It is an exit checking loop. In do...while loop the test condition is given at the end of the loop. Therefore the body of the loop will be executed at least once. If the test condition is true, then repeat the body of the loop otherwise exit from loop.

Syntax:

```
do
{
Body of loop statements
} while (test expression);
```

Flowchart:



Program: Print the Sum of the Series 1 + 2 + 3 + 4 . . . up to N terms

```
#include<stdio.h>
#include<conio.h>
void main ( )
```

```
{
int i, sum = 0, n;
printf ("Enter the number of terms");
scanf ("%d", &n);
i=1;
do
{
sum = sum+ i;
i++;
} while( i <= n);
printf (" Sum = %d", sum);
getch();
}
```

Output

Enter the number of terms 5

Sum=15

Difference Between while and do . . . while

while	do….while
Condition is tested at the beginning of the loop	Condition is tested at the end of the loop
Some time the loop will not be executed at least once.	Loop will be executed at least once even though the condition is false.