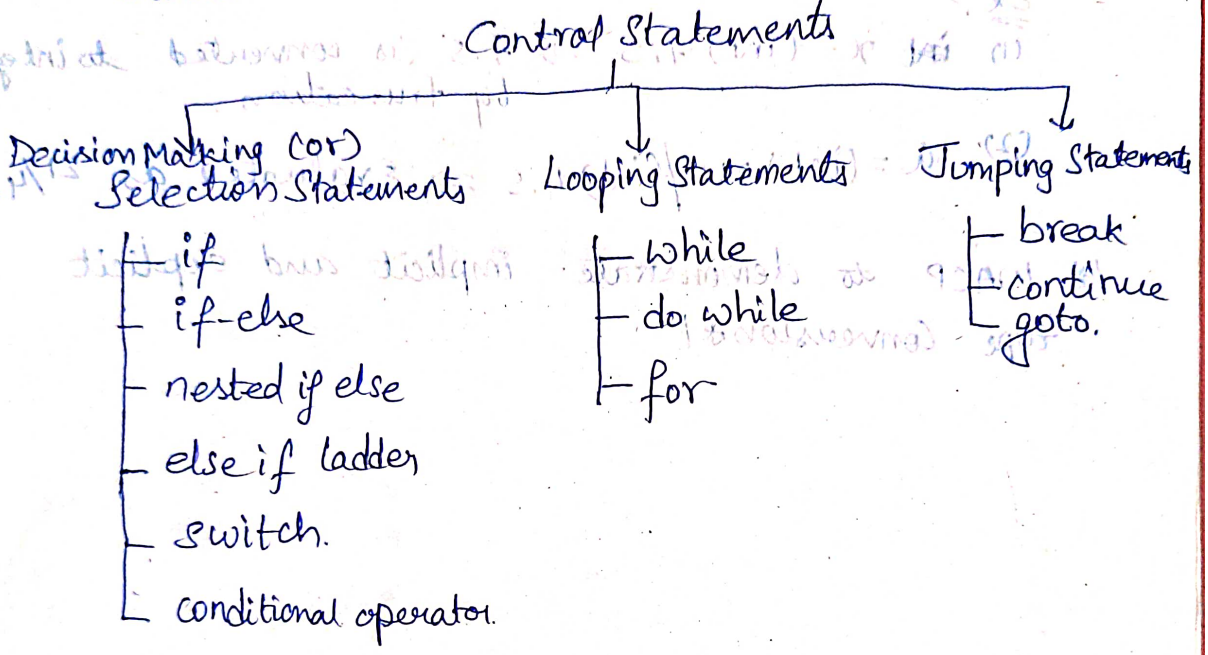
→ CONTROL STATEMENTS (Conditional Branching and Loops)

Control statements enable us to specify the flow of program control i.e., the order in which the instructions in a program must be executed.



(1) Decision Making (or) Selection statements:-

These statements decide the order of execution of statements based on conditions. There include

(i) Simple if statement:-

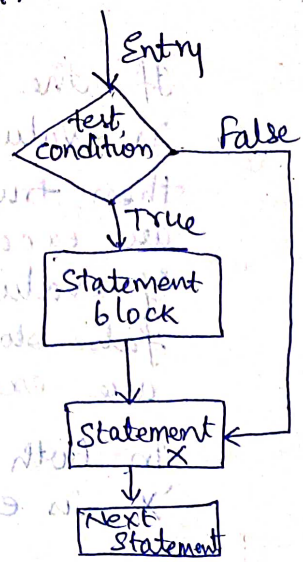
The 'if' statement is a two-way decision statement and is used with a condition.

Syntax:

```
if (condition)
  {
     Statement block;
   // Executes if condition is true.
  }
Statement-X;
```

VSLK

Flow Chart:-



If the test condition is true, the statement block gets executed; otherwise Statement-X gets executed.

In both true or false cases, statement-X gets executed.

```
/* WACP to demonstrate simple-if */
# include <stdio.h>
void main()
{
   int a=10;
   if(a==10)
   {
      printf("Hello");
   }
   printf("\n Task Complete");
}
```

Output:-

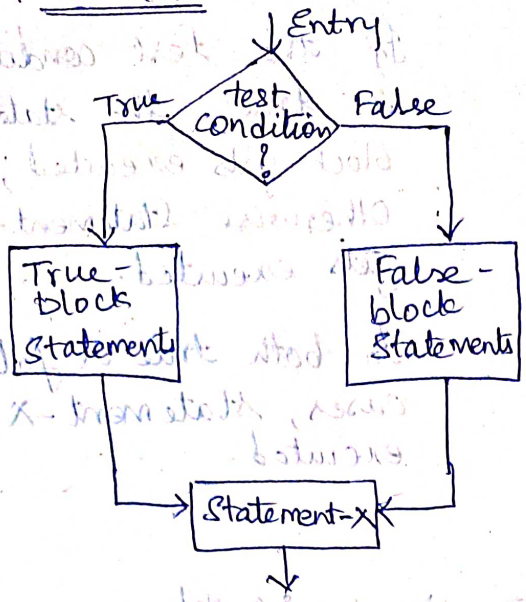
```
./a.out
Hello
Task Complete.
```

(ii) if-else statement:-

if..else statement is an extension of simple if statement. The Syntax/general form:-

```
if (test condition)
  {
    True-block statements;
  }
else
  {
    False-block statements;
  }
Statement-X;
```

Flowchart



If the test condition is evaluated to true, then true block statements are executed, if evaluated to false, false block statements are executed.

In both case, statement 'x' is executed.

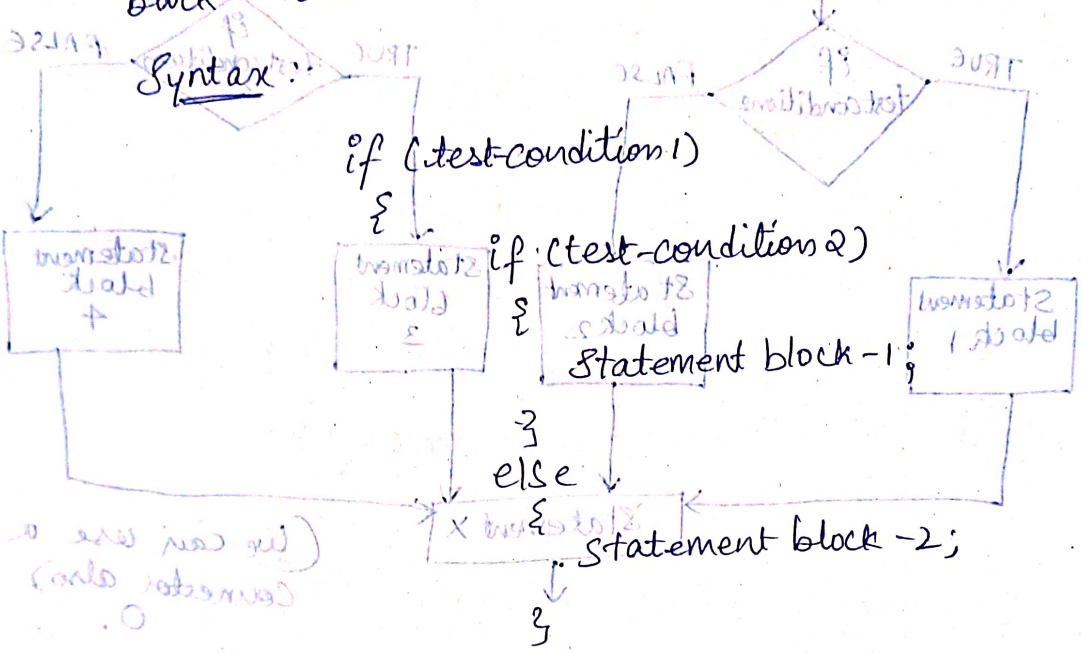
```
/* WACP to demonstrate if-else */
#include<stdio.h>
Void main()
{
  int a,b;
  printf("\n Enter 2 values");
  scanf("%d%d",&a,&b);
  if(a>b)
  {
    printf("%d is greatest",a);
  }
  else
  {
    printf("%d is greatest",b);
  }
  printf(" Found greatest number!!!");
}
```

Output:

```
o/a.out
Enter 2 values
5
2
5 is greatest
Found greatest number!!!
o/a.out
Enter 2 values
6
10
10 is greatest
Found greatest number
```

(iii) nested if-else:-

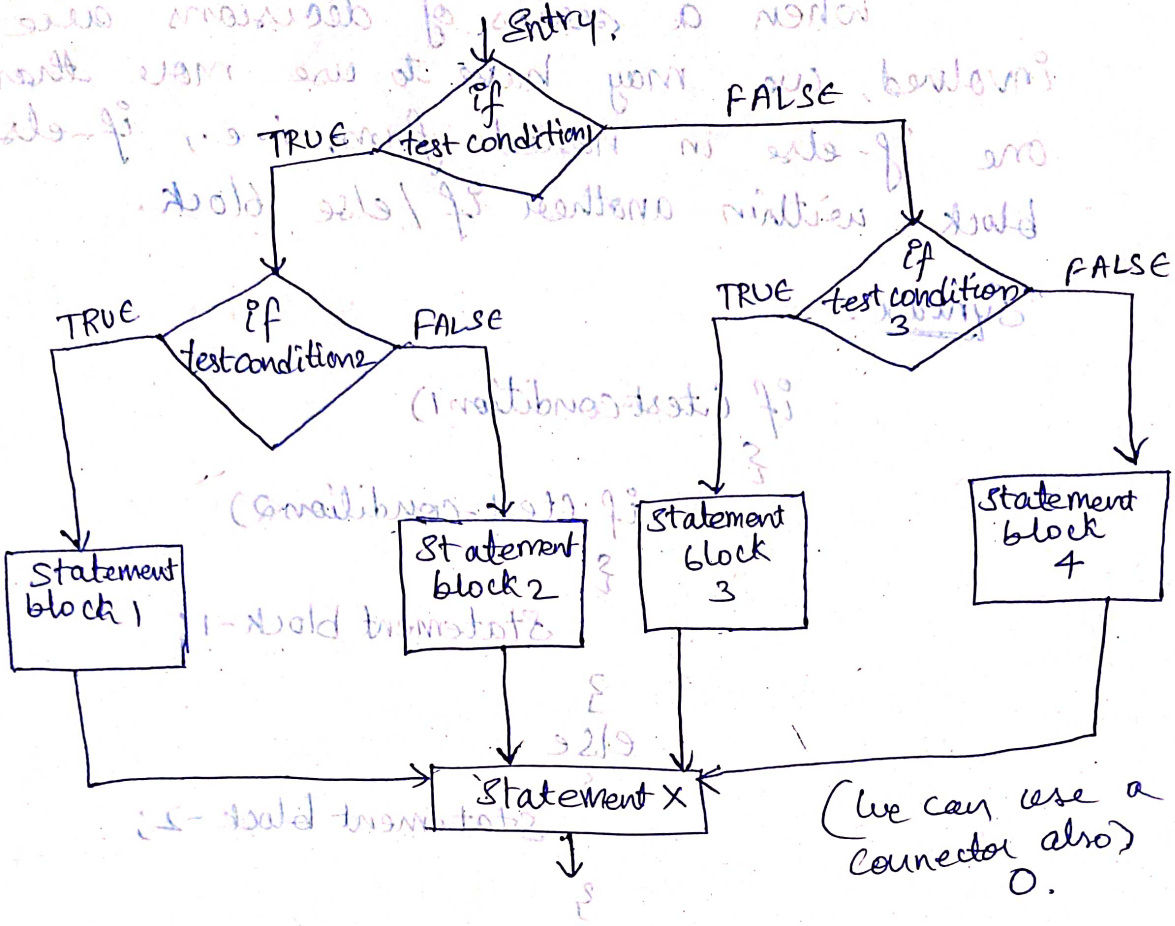
When a series of decisions are involved, we may have to use more than one if-else in nested form, i.e., if-else block within another if/else block.



Syntax:-

```
if (test-condition 1)
{
    if (test-condition 2)
    {
        Statement block-1;
    }
    else
    {
        Statement block-2;
    }
}
else
{
    if (test-condition 3)
    {
        Statement-block-3;
    }
    else
    {
        Statement block-4;
    }
}
```

Flowchart :-



If the test condition 1 and 2 are true then only statement block 1 will be executed.

Statement-X gets executed after any of the statement block gets executed. ~~or none~~

```
/* WACP to find largest of 3 numbers using
   nested if-else */
#include <stdio.h>
void main()
{
    int a, b, c;
    printf("\n Enter any 3 numbers:");
    scanf("%d%d%d", &a, &b, &c);
```

```
if (a>=b)
{
  if (a>=c)
  {
    printf("\n %d is largest", a);
  }
  else
  {
    printf("\n %d is largest", c);
  }
}
else
{
  if (b>=c)
  {
    printf("\n %d is largest", b);
  }
  else
  {
    printf("\n %d is largest", c);
  }
}
```

Output :- % a.out

Enter any 3 numbers

5

9

4

9 is largest

(iv) else-if ladder:-

else-if ladder is used 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 (condition 1)
    {
      statement block-1;
    }
else if (condition 2)
    {
      statement block-2;
    }
else if (condition 3)
    {
      statement block-3;
    }
    .
    .
    .
else if (condition n)
    {
      statement block-n;
    }
else
    {
      default-statement;
    }
statement-x;
```

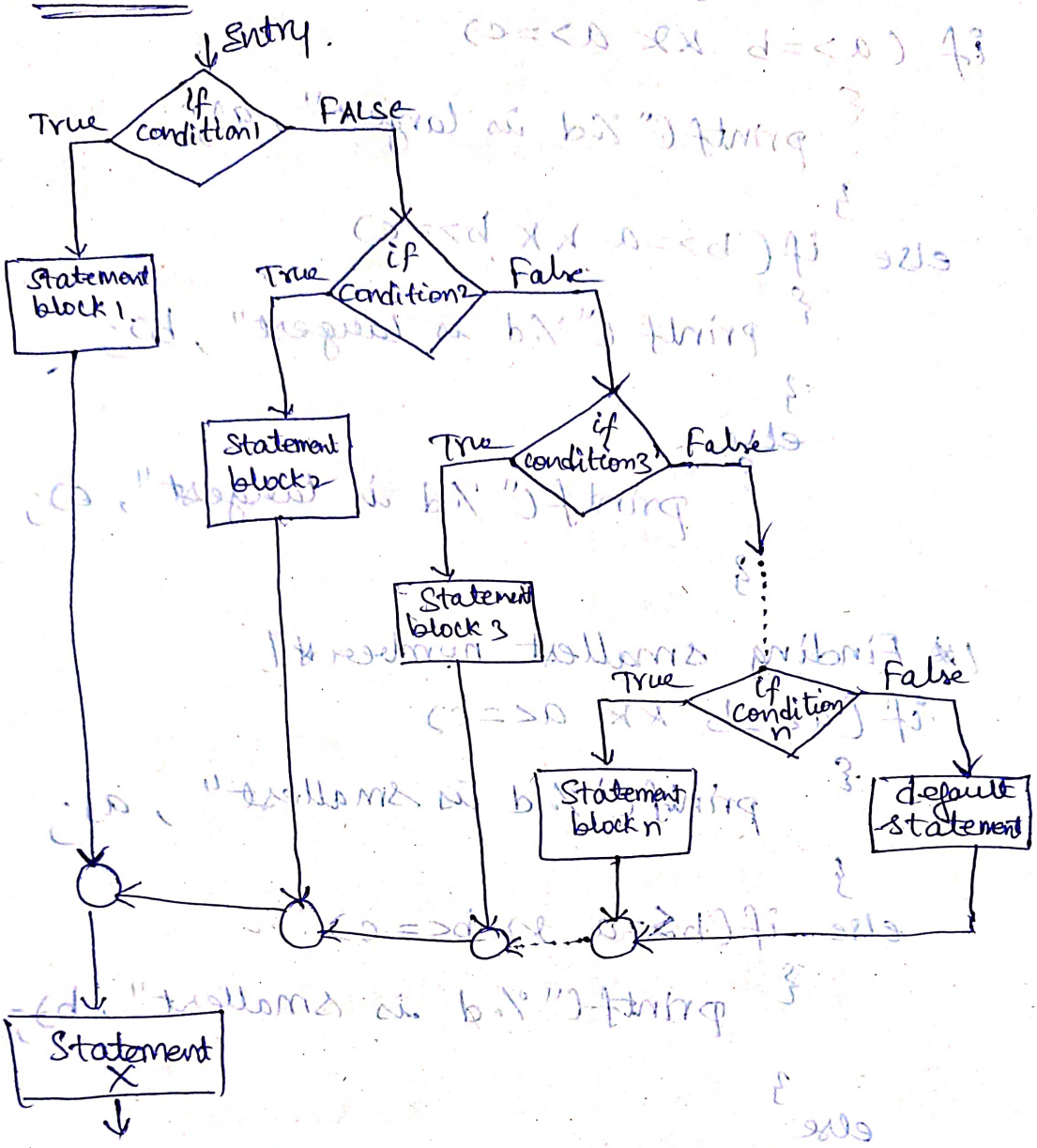
This construct is known as else-if ladder. The conditions are evaluated from top of ladder to downwards.

As soon as a true condition is found, the statement associated with it is executed and control is transferred to statement-x. (skipping the rest of the ladder).

When all the n conditions become false, then final else containing default statement gets executed.

(VSLK) (28)

Flowchart :-



~~/* incto~~ /* WACP to find largest and smallest of 3 numbers */

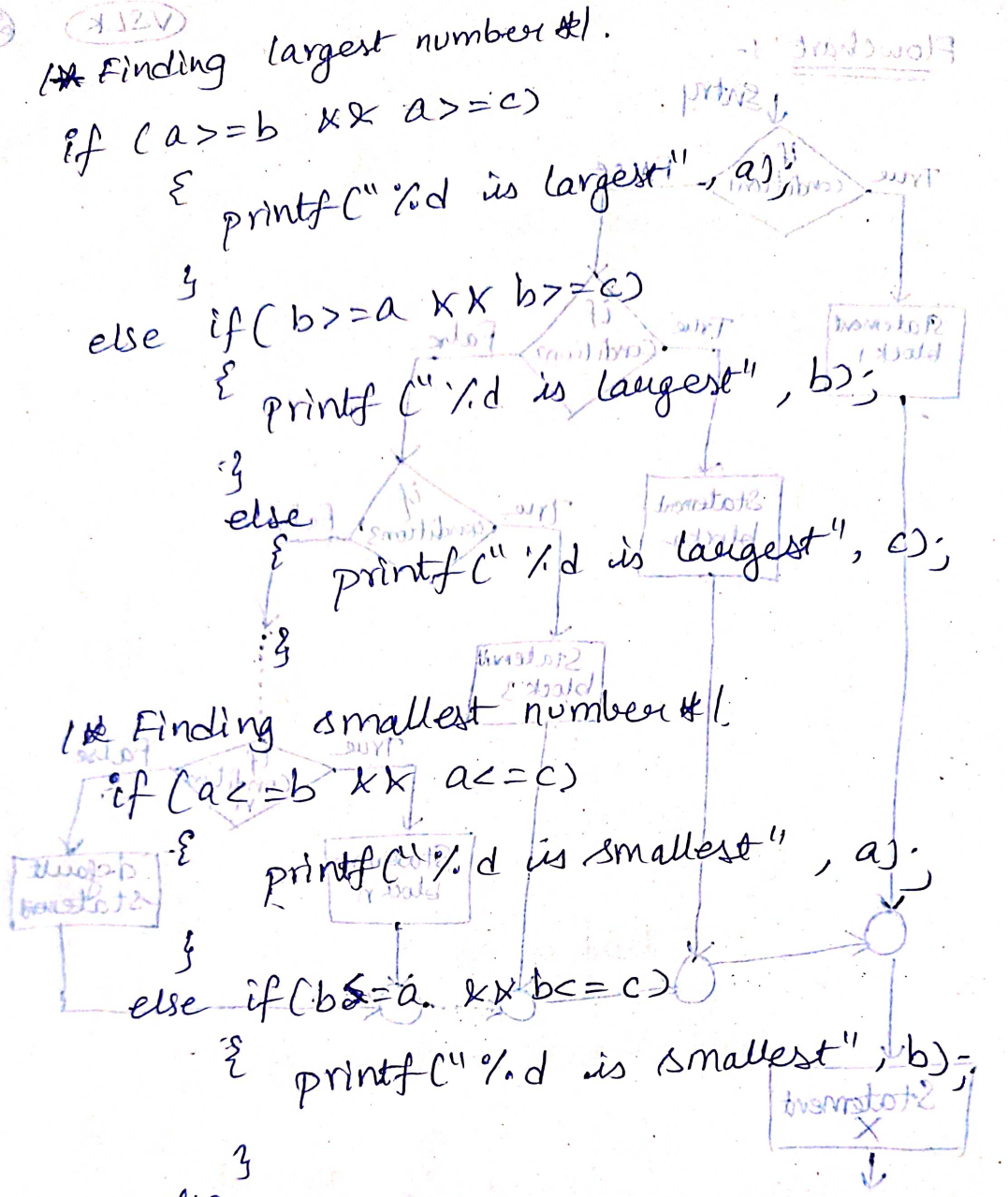
```
#include<stdio.h>
void main()
{
    int a,b,c;
    printf("\n Enter any 3 numbers: ");
    scanf("%d %d %d", &a, &b, &c);
```

/* Finding largest number */

```
if (a>=b && a>=c)
  {
    printf("%d is largest", a);
  }
else if(b>=a && b>=c)
  {
    printf("%d is largest", b);
  }
else
  {
    printf("%d is largest", c);
  }
```

/* Finding smallest number */

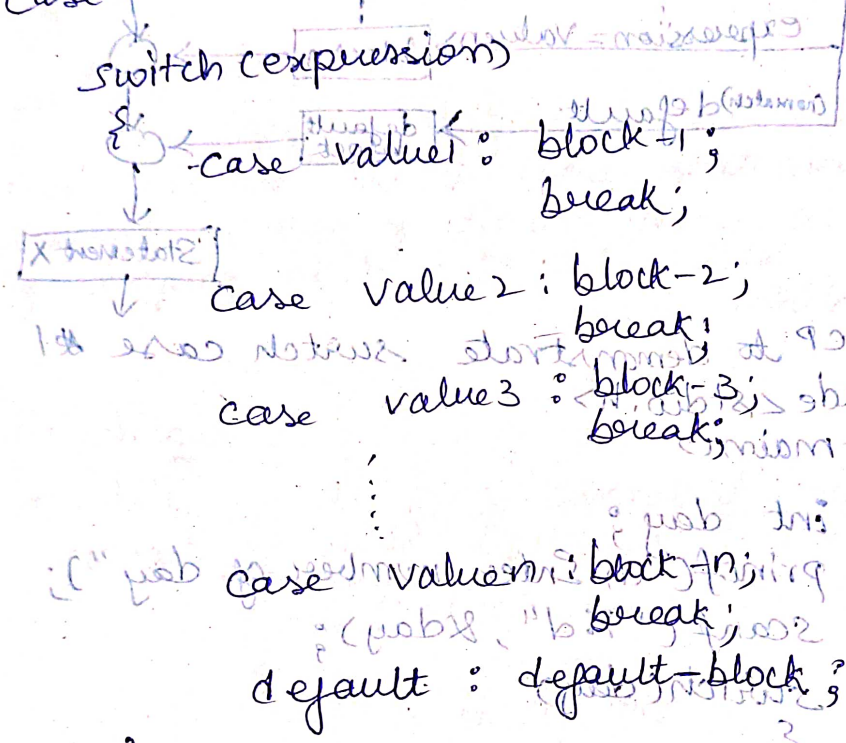
```
if (a<=b && a<=c)
  {
    printf("%d is smallest", a);
  }
else if(b<=a && b<=c)
  {
    printf("%d is smallest", b);
  }
else
  {
    printf("%d is smallest", c);
  }
}.
```



VSLK (2a)

(v) The Switch statement :

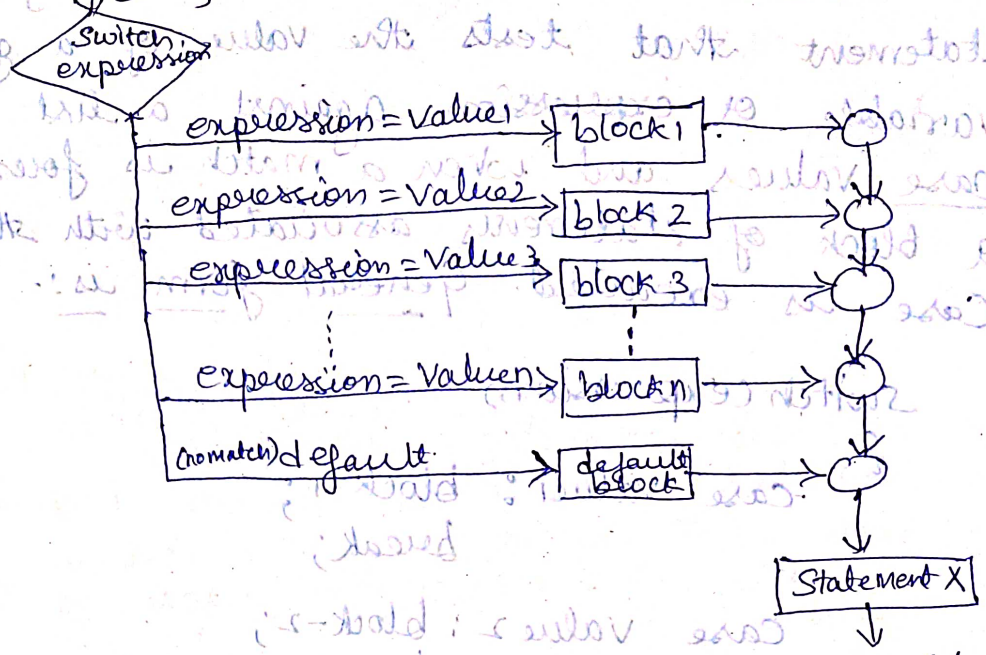
The switch statement is a multiway decision statement that tests the value of a given variable or expression against a list of case values and when a match is found, a block of statements associated with that case is executed. General form is :-



```
switch (expression)
{
    case value1 : block-1;
                  break;
    case value2 : block-2;
                  break;
    case value3 : block-3;
                  break;
    .
    .
    .
    case valuen : block-n;
                  break;
    default : default-block;
}
statement-x;
```

* the expression should be an integer expression or character
* value1, value2, ... valuen are constants and are known as case labels. They are unique.
* The break statement at the end of each block marks the end of a particular case and causes exit from switch if the block gets executed.

Flowchart:




```
/* WAP to demonstrate switch case */
#include<stdio.h>
void main()
{
    int day;
    printf("\n Enter number of day ");
    scanf("%d", &day);
    switch(day)
    {
        case 1 : printf("\n Sunday"); break;
        case 2 : printf("\n Monday"); break;
        case 3 : printf("\n Tuesday"); break;
        case 4 : printf("\n Wednesday"); break;
        case 5 : printf("\n Thursday"); break;
        case 6 : printf("\n Friday"); break;
        case 7 : printf("\n Saturday"); break;
        default : printf("\n enter other than 1-7");
    }
    printf("\n Switch execution complete.");
}
```

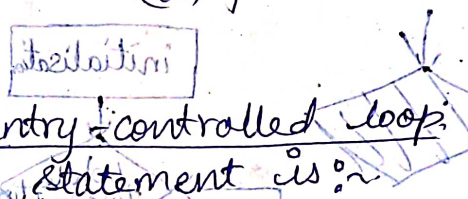
Output:

```
./a.out
Enter number of day : 5
Thursday. Switch execution complete
./a.out
Enter number of day : 10.
U entered other than 1-7
Switch execution complete
```

(9) Looping statements:

Looping statements are the statements that execute one or more statements repeatedly several number of times. There are 3 types of looping statements :- (i) while
(ii) do-while
(iii) for

(i) while :-



* while is an entry-controlled loop. The basic format of while statement is :-

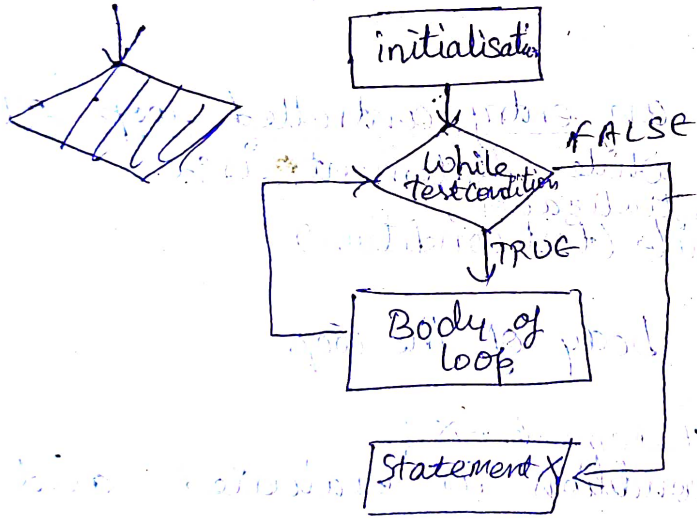
```
initialization;
while (test condition)
{
    body of the loop
}
statement-x;
```

* the test condition is evaluated and if the condition is true, then body of loop gets executed. This process of repeated execution of body continues until the test-condition becomes false.
* This is also called counter-controlled loop. The counter must be initialized, tested and updated properly for desired operations.

```
/* WACP to demonstrate while loop */
#include<stdio.h>
void main()
{
   int i;
   for(i=1;i<=3;
   i=1;        //initialization
   while (i<=3) // condition checking.
   {
      printf("\n Hello World");
      i=i+1; // Incrementing i.
   }
}
```

Output:-
Hello World
Hello World
Hello World

$ Flowchart:-



(ii) do-while loop:-

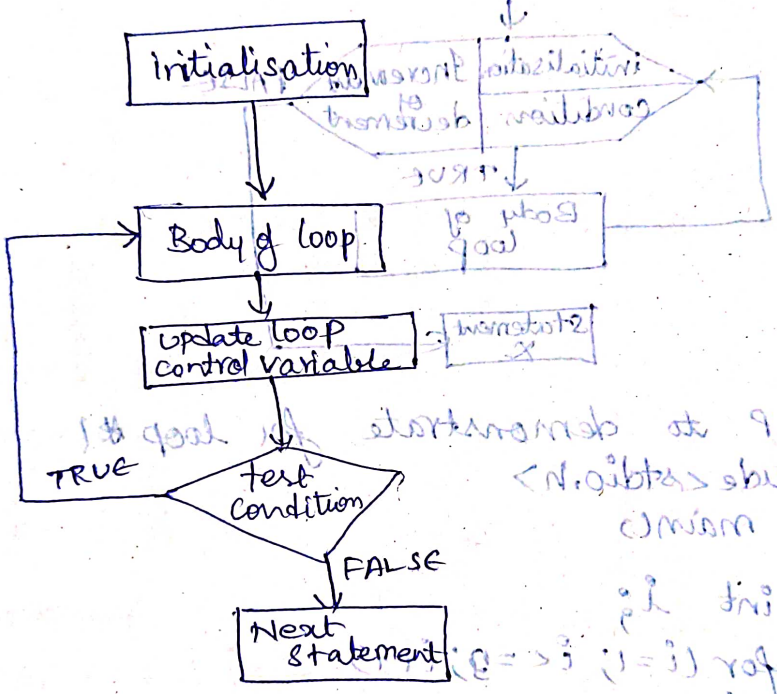
do-while is an exit controlled loop.

On reaching the do statement, the program proceeds to evaluate the body of loop first. At the end of the loop, the test condition in the while statement is evaluated.

Syntax:-

```
initialization;
do
{
    Body of the loop
    Update loop control variable.
} while (test condition);
```

Flowchart:-



/* WAP to demonstrate do-while loop */

```
#include<stdio.h>
void main()
{
    int i;
    i=1; // initialization
    do
    {
        printf("\n Hello");
        i=i+1;
    } while(i<=3);
}
```

Output:-

Hello world
Hello world
Hello world.

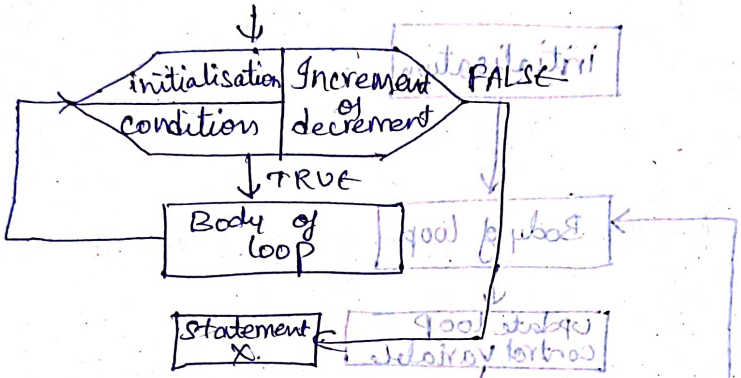
(iii) **for loop:-**

'for' loop is an entry controlled loop where the initialisation, condition checking and update control variable are written in same statement

Syntax:

```
for(initialisation; testcondition; updation/increment/decrement)
{
    Body of loop
}
```

Flowchart:



WAP to demonstrate for loop #1

```
#include<stdio.h>
void main()
{
    int i;
    for(i=1; i<=3; i++)
    {
        printf("\n Hello World");
    }
    printf("\n After for loop");
}
```

Output:

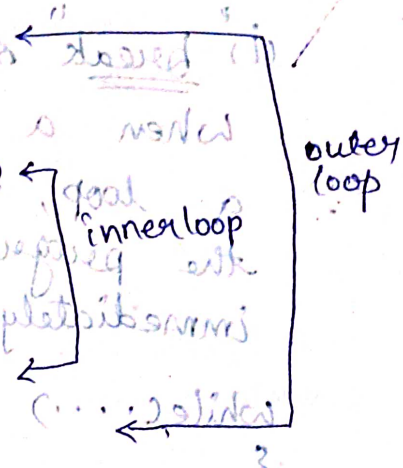
```
./a.out
Hello World
Hello World
Hello World
After for loop.
```

Nesting of loops:-

loop within another loop is called nested loop

General form:-

```
for(i=1; i<=10; i++)
{
    for(j=1; j<=10; j++)
    {
    }
}
```



/* WAP to print * * * using nested for loops */

```
#include<stdio.h>
void main()
{
    int i,j;
    for(i=1; i<=3; i++)
    {
        for(j=1; j<=3; j++)
        {
            printf("* ");
        }
        printf("\n");
    }
}
```

Output

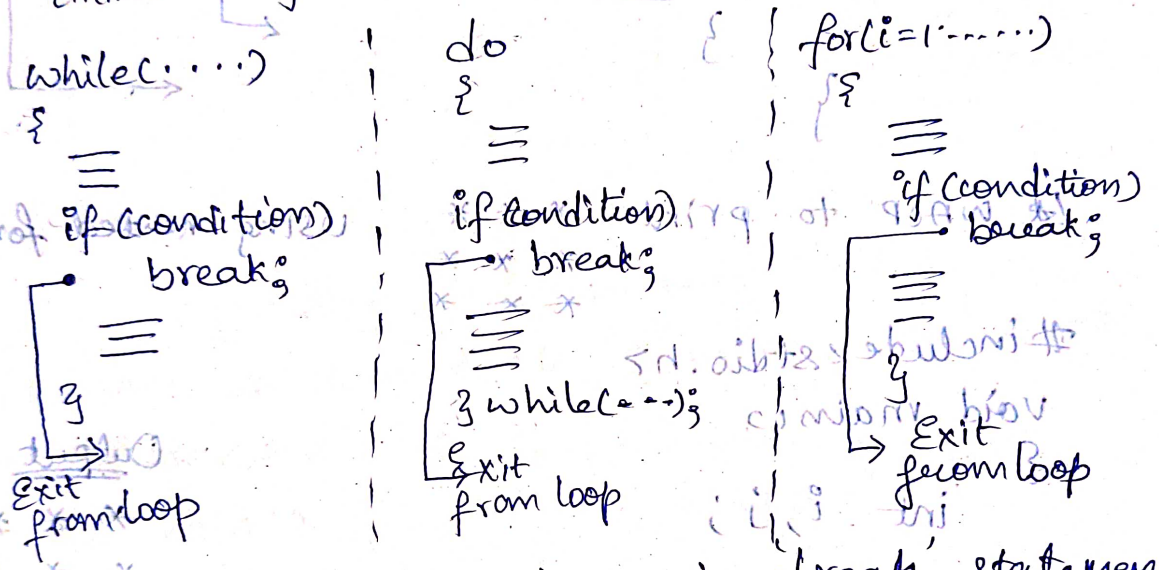
```
* * *
* * *
* * *
```

(3) Jumping Statements :-

Jumping statements permit a jump from one statement to another within a loop as well as a jump out of loop.

(i) "break" statement :-

When a break statement is executed in a loop, the loop is immediately exited and the program continues with the statement immediately after the loop.



Exiting from loop using break statement

```
/* WAP to demonstrate 'break' statement */
#include<stdio.h>
void main()
{
    int i;
    for(i=1; i<=10; i++)
    {
        if(i==5)
            break;
        printf("%d", i);
    }
}
```

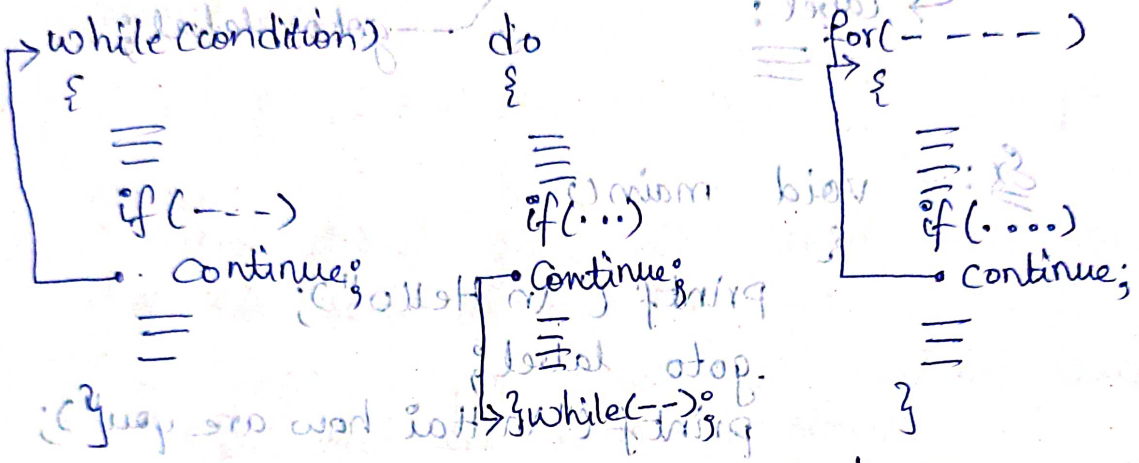
Output :-

```
1
2
3
4
```

VSLK (33)

(ii) "continue" statement:

'continue' statement skips a part of the loop, by causing the loop to be continued with next iteration after skipping any statements in between.



Bypassing and continuing in loops.

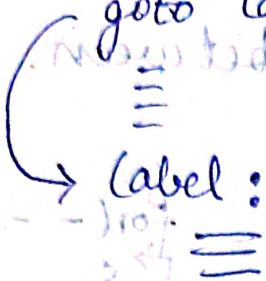
```
/* WACP to demonstrate continue statement */
#include<stdio.h>
void main()
{
  int i;
  for(i=1; i<=10; i++)
  {
    if(i==5)
      continue;
    printf("%d", i);
  }
}
```

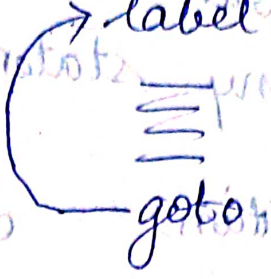
Output :-

1
2
3
4
6
7
8
9
10

(iii) goto statement :-

The goto statement is an unconditional jump statement used to jump from anywhere to anywhere within a program.





Ex :-

```
void main()
{
    printf("\n Hello");
    goto label;
    printf("\n Hai how are you");
    label:
    Printf(" World");
}
```

Output :- Hello World //