Chaithra H.E

Module 2 :- (continuation) (part 2)

Branching & Looping :-

2 way selection :- if
if-else
nested if-else
switch statement
goto statement
} decision making statements

Loops : for
while
do-while
break and continue (unconditional branching statement)
} Loops

Simple Statement :-

-only one statement encloseds within pair of braces such as '{' and '}'.

Compound statement :

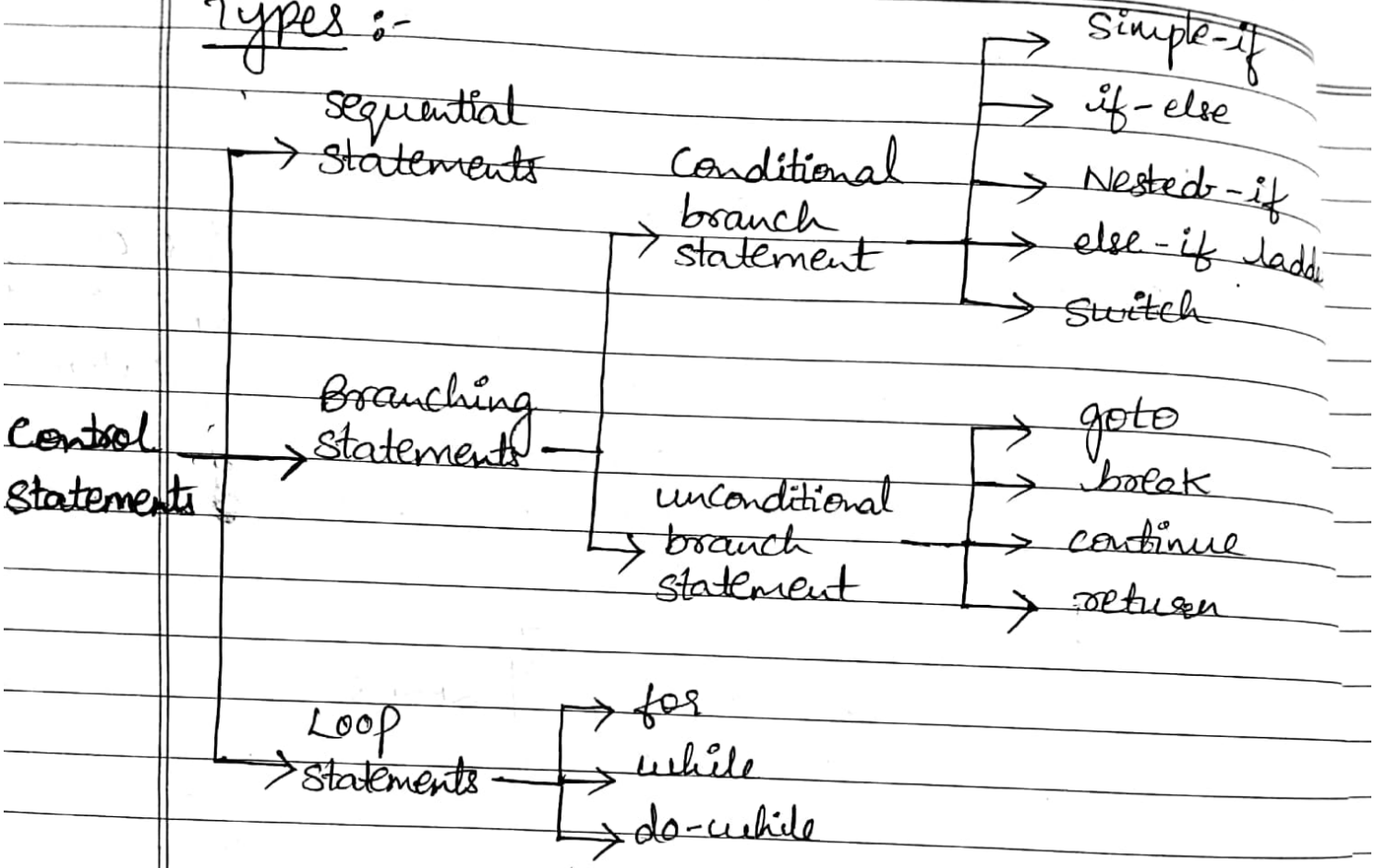
Set of statements encloseds within pair of braces { and }.

Control Statements :-

The statements that are used to control the flow of execution of program is called control statement.

By, Chaithra H.E.

Types :-



* Sequential Statements

The programming statements that are executed sequentially, one after the other are called as sequential statements.

```
Statement -1;
statement -2;
    :
    :
Statement n;
```

* Branching statements :-

The statement which alters the sequence of execution of program are called branching statements.

- Conditional branch statement

The statements that alter the sequence of execution of program based on the condition.

It is also called as decision statement

→ if
→ if-else
→ Nested if-else
→ else-if ladder
→ switch

1) **if :-** if statement is a decision statement which is used to execute statement or a set of statements conditionally.
- It is a one-way decision statement
- Here condition is tested which results in either a true or a false value.

- If the condition is true then the statement which is inside the body of if will be executed.

- If the condition is false then the next statement which is executable i.e., outside the body of if will be executed.

Syntax:-
```
if (condition)
{
  statement;
}
```


condition → Logical expression that results in true or false.

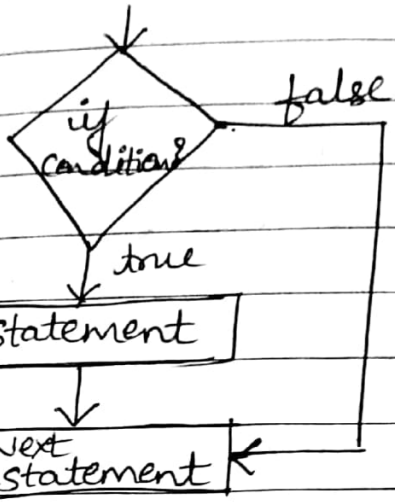
statement → simple or compound statement

simple statement : single statement

compound statement : set of statements.

Note :- Braces are not required for simple statement.

Flowchart :-



Example1:- WAP to accept a number & check whether it is even number & print the same.

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

output :- Enter a number
10
number is even

Example 2:- WAP to find the largest of 2 numbers using if statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a, b, big;
  clrscr();
printf("Enter 2 numbers\n");
 scanf("%d%d",&a,&b);
 big=a;      // assigning a to big
 if(b>big).
 big=b;
 printf("Largest number is %d", big);
 getch();
}
```

Output :-

Enter 2 numbers
10 20
Largest number is 20.

2) if-else :-

It is a 2-way decision statement. i.e., one set of statement will be executed when the condition is true & another set of statement will be executed when the condition is false.

Syntax :-

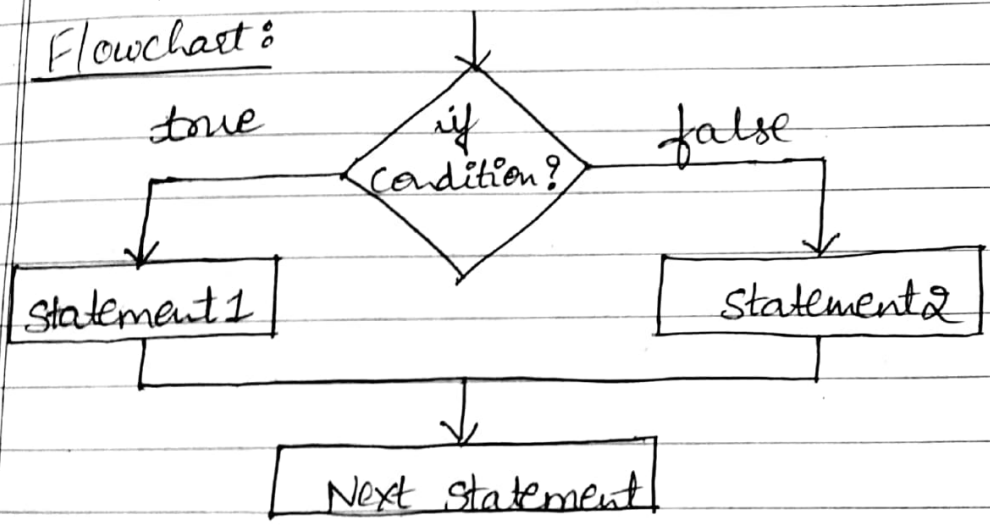
```
if (condition)
{
  statement 1;
}
else
{
  statement 2;
}
```

CHAITHRA. H.E.
B.E, M. Tech

The condition is tested. If the condition is true, then statement 1 is executed. otherwise, statement 2 is executed.

The statement 1 and statement 2 may be simple or compound statements.

Flowchart:



Example 1:- WAP to check whether number is even or odd.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n;
```

```
printf("Enter a number\n");
scanf("%d", &n);
if(n%2==0)
{
  printf("even number");
}
else
{
  printf("odd number");
}
getch();
}
```

Output

Enter a number

5

odd number.

Lab program

1) a) Design & develop a C program to read a year as an input and find whether it is leap or not. (Also consider end of centuries).

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int year;
  clrscr();
  printf("enter the year\n");
```

```
scanf("%d", &year);
if((year%4==0)&&(year%100!=0)||(year%400==0))
{
printf("%d is a leap year", year);
}
else
{
printf("%d is not a leap year", year);
}
getch();
}
```

Example 2: WAP to find the largest of 2 numbers using if-else.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a, b;
clrscr();
printf("enter a and b values\n");
scanf("%d%d", &a, &b);
if(a>b)
{
printf("a is greater");
}
else
{
printf("b is greater");
}
```

```
    getch();
}
```

3) nested-if :-

An if or if-else statement enclosed within another if or if-else statement is called nested-if statement.

An action is performed based on many decision this is also called as multi-way decision statement.

Syntax :-

```
if (condition 1)
{
  if (condition 2)
  {
    statement 1;
  }
  else
  {
    statement 2;
  }
}
else
{
  if (condition 3)
  {
    statement 3;
  }
```

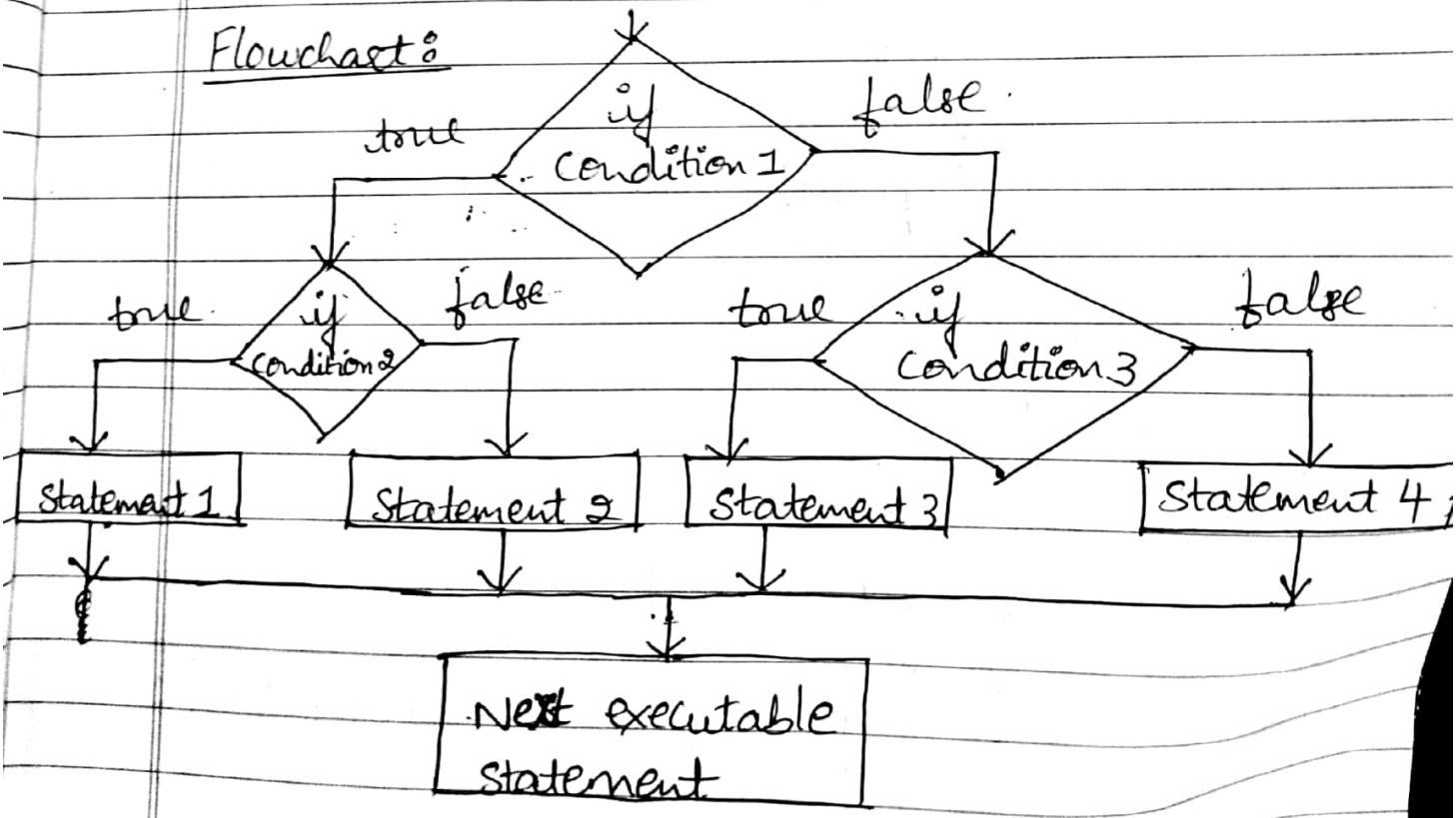
```
else
{   statement 4;

}
}
```

In the above syntax,

* Statement 1 is executed if condition 1 & condition 2 are true.
* If the condition 1 is true & condition 2 is false then statement 2 is executed.
* If the condition 1 is false then control transfers to the else part.
* In the else part if the condition 3 is true then statement 3 is executed, else statement 4 is executed.

Flowchart:



Example: WAP to find the largest of 3 numbers using nested-if

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a, b, c;
  clrscr();
  printf("enter a, b and c values\n");
  scanf("%d%d%d", &a, &b, &c);
  if(a>b)
  {
    if(a>c)
    {
      printf("a is largest\n");
    }
    else
    {
      printf("c is largest\n");
    }
  }
  else
  {
    if(b>c)
    {
      printf("b is largest\n");
    }
    else
    {
```

```
        printf("c is largest\n");
    }
}
getch();
}
```

4) else-if ladder :-

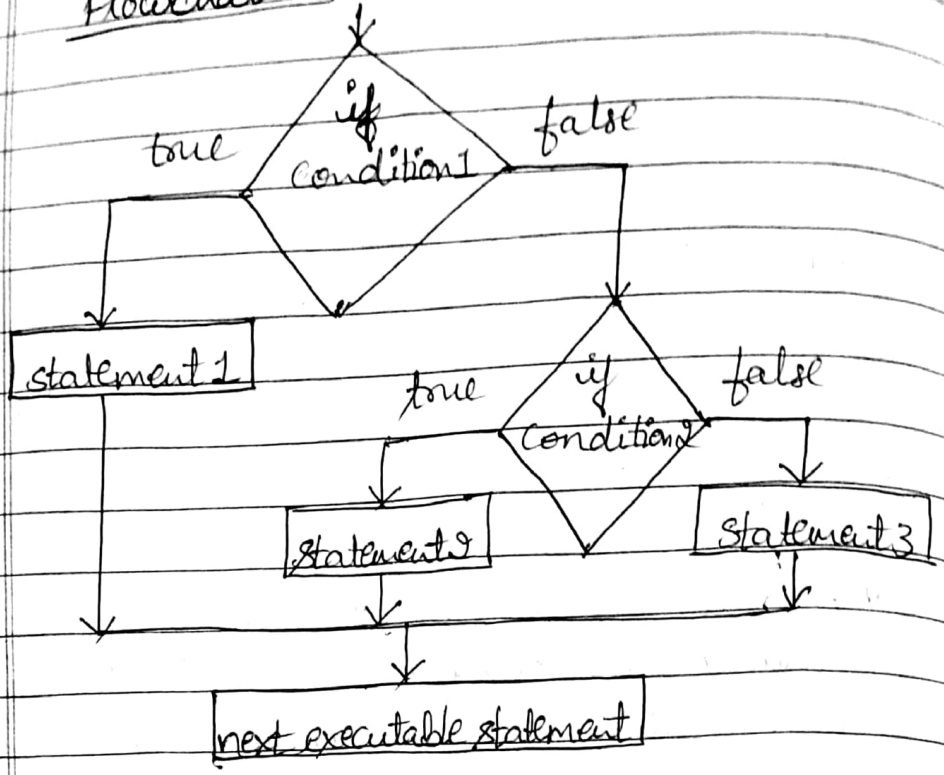
when an action is to be performed based on many decisions, then this statement is used. So it is called multi-way decision statement.

It is a form of nested-if statement where nesting is allowed only in the else part.

Syntax :-

```
if (condition1)
{
    statement1;
}
else if (condition2)
{
    statement2;
}
else
{
    statement3;
}
```

Flowchart :-



Example :- WAP to check whether a number is positive, negative or zero.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n;
  clrscr();
  printf("enter a number\n");
  scanf("%d",&n);
  if(n>0)
  {
   printf("number is positive");
  }
  else if(n<0)
  {
   printf("number is negative");
  }
```

```
else
{
  printf("number is zero");
}
getch();
}
```

5) Switch Statement :-

Switch statement provides a multiple way of branching.

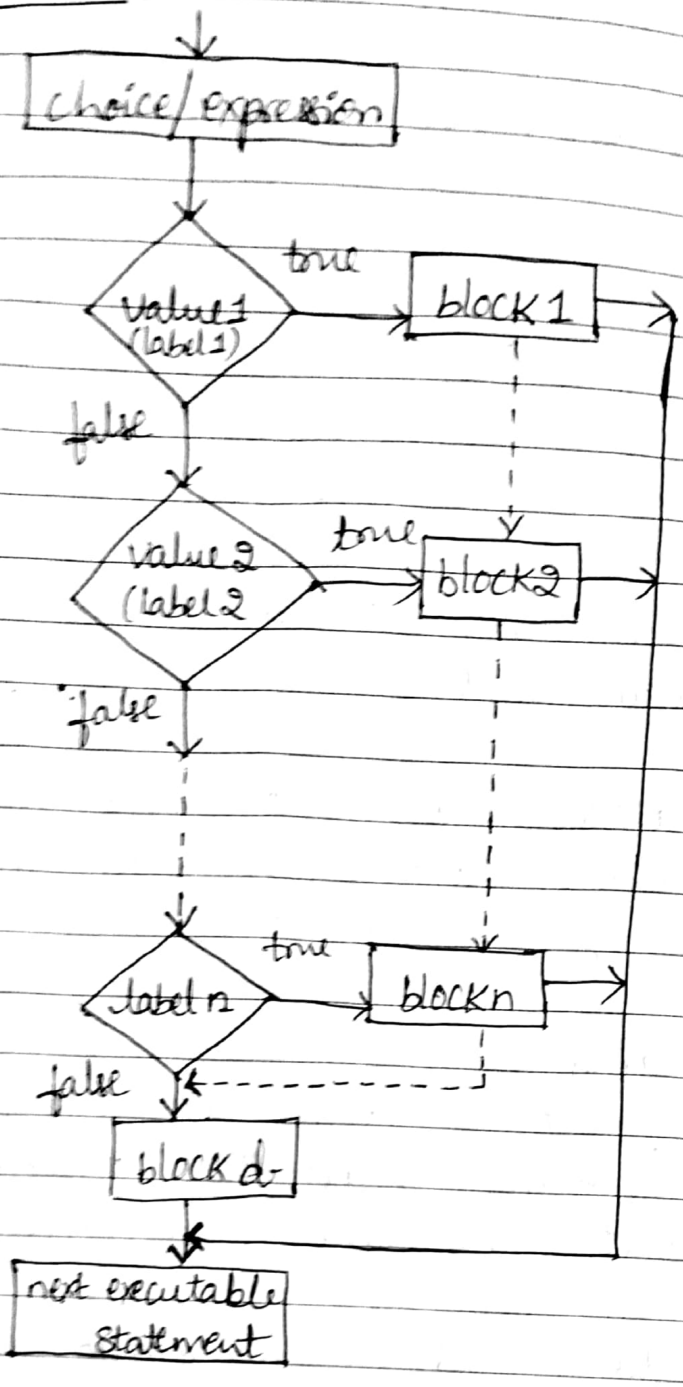
It allows user to select any one of the several alternatives, depending on the value of expression (choice).

Depending upon the value of expression, the control transfers to a particular case label (branch) & all the statements followed by that case label are executed.

Syntax :-

```
switch(choice/expression)
{
  case label1 : block1;
                break;
  case label2 : block2;
                break;
        .
        .
        .
  case labeln : blockn;
                break;
  default : blockd;
}
```

Flowchart :-



Example :- WAP to simulate the functions of a simple calculator.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    float a, b, res;
    char op;
    clrscr();
```

```
printf("Enter the expression\n");
scanf("%f %c %f", &a, &op, &b);
switch(op)
{
case '+': res=a+b;
          break;

case '-': res=a-b;
          break;

case '*': res=a*b;
          break;

case '/': if(b!=0)
          {
          res=a/b;
          }
          else
          {
          printf("Div. by zero\n");
          exit(0);
          }
          break;
default : printf("Illegal operator\n");
          exit(0);

}
printf("%f %c %f = %f\n", a, op, b, res);
}
```

Note :- exit(0) is successful termination from main function.

Break statement prevents the code running into the next case.

6) goto statement :-

By using goto statement, the control of a program can be transferred from one statement to the specified statement without any condition.

Syntax :- goto label;

:

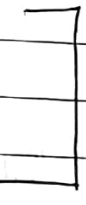
label: statement;

where, goto is a keyword
label is any identifier.

when goto statement is encountered the control is transferred to the label after which statement is executed.

a)

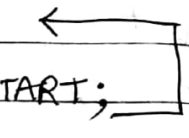
```
goto END;
:
END:
```



Here, The statements immediately followed by the goto will be skipped, while control jumps to label END. This type is known as a forward jump.

b)

```
START:
:
goto START;
```



Here, the statement inside label start will be executed. the label START: is placed before the goto statement. This type is known as a backward jump.

Example :- WAP to find sum of 10 natural numbers using goto statement.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i, sum;
  clrscr();
  i=0;
  sum=0;
top: sum=sum+i;
     i++;
     if(i<=10)
goto top;
printf("sum=%d\n", sum);
}
```

WAP to find whether the number is armstrong or not.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
  int n, d1, d2, d3, sum;
```

```
clrscr();
printf("Enter the value of n\n");
scanf("%d", &n);
d3 = n%10;
d2 = (n/10)%10;
d1 = n/100;
sum = pow(d1,3) + pow(d2,3) + pow(d3,3);
if(sum == n)
printf("Armstrong number");
else
printf("Not an armstrong number");
}
```

Loops :-

while
do while
for

1) while :- (pre-testing loop)

*A set of statements may have to be repeatedly executed till a certain condition is reached. So we use while loop.

* while is a loop where set of statements will be executed repeatedly as long as the condition is true. When the condition becomes false then control of program will terminate the while loop.

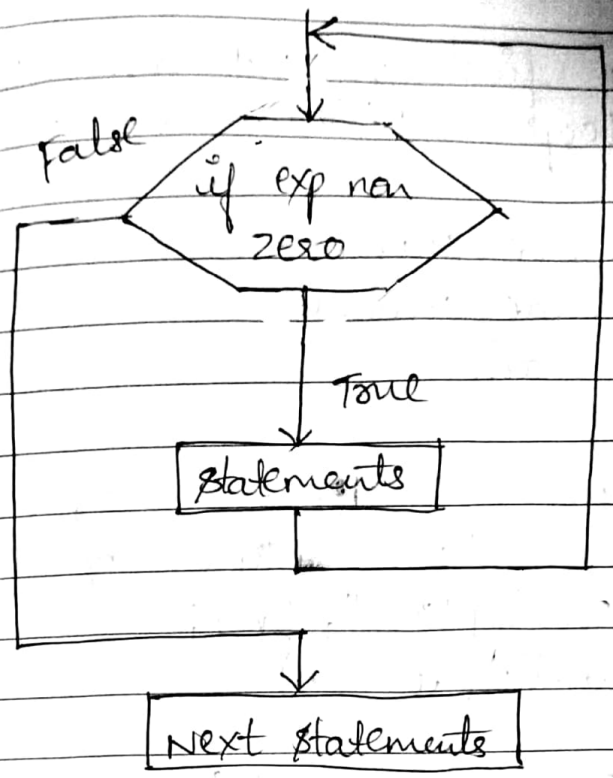
Syntax :-

```
while(exp)
{
    statements;
}
```

CHAITHRA. H.E.
B.E, M. Tech

where, while is a keyword.
exp is the expression which is evaluated to TRUE or False.

Flowchart :-



1) WAP to accept name & print it for a specified number of times.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char name[20];
  int ntimes, i=1;
  clrscr();
  printf("enter name\n");
  scanf("%s", name);
  printf("How many times to print name\n");
  scanf("%d", &ntimes);
  while(i<=ntimes)
  {
    printf("%s", name);
    i++;
  }
```

```
    getch();
}
```

2) WAP to find sum of given series using while loop. (1+2+3+4+----+n)

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, i=1, sum=0;
    printf("Enter number of elements\n");
    scanf("%d", &n);
    while(i<=n)
    {
        sum = sum+i;
        i++;
    }
printf("Sum of series = %d", sum);
getch();
}
```

3) WAP to compute sum of series $1+x^1+x^2+---+x^n$.

($1 \rightarrow x^0$)

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n, i=0, sum=0, x;
    clrscr();
    printf("Enter value of x and n\n");
```

```
scanf("%d%d", &x, &n);
while(i<=n)
{
    sum = sum + pow(x, i);
    i++;
}
printf("Sum of series = %d", sum);
getch();
}
```

4) WAP to compute GCD & LCM of 2 numbers

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int m, n, p, q, gcd=0, lcm=0;
    clrscr();
    printf("Enter 2 numbers\n");
    scanf("%d%d", &m, &n);
    p=m;
    q=n;
    while(n!=0)
    {
        r=m%n;
        m=n;
        n=r;
    }
    gcd=m;
    lcm=(p*q)/gcd;
    printf("gcd=%d", gcd);
    printf("lcm=%d", lcm);
    getch();
}
```

Note:- Refer text books for more example programs.

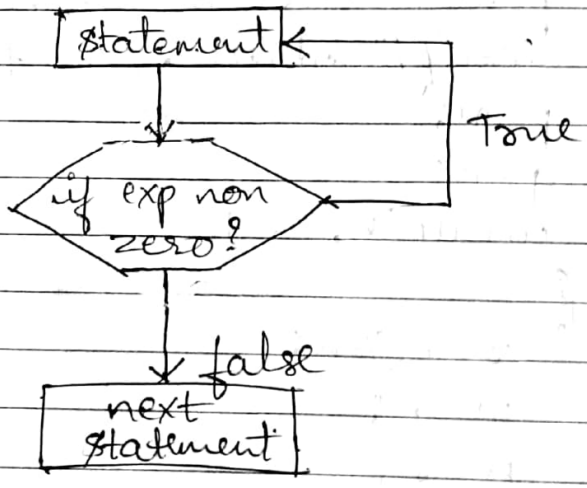
2) do-while :- (post testing loop)

* It is a loop where set of statements will be executed repeatedly as long as the condition is true.
* It is a post-testing loop where code will be executed atleast once, then condition will be checked.

Syntax :-

```
do
{
    statements;
}while(exp);
```

Flowchart :-



1) WAP to ~~convert~~ find sum of first n integers.

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

```
{
int n, i=01, sum=0;
clrscr();
printf("enter n\n");
scanf("%d",&n);
do
{
sum=sum+i;
i++;
}while(i<=n);
printf("sum=%d",sum);
getch();
}
```

3) for loop :-

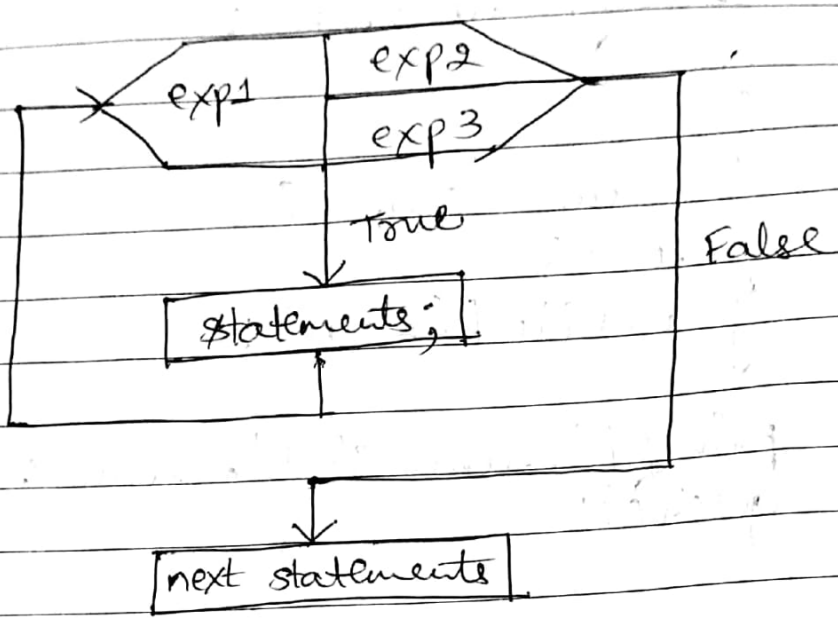
* Loop in which code will be executed for a specified number of times.

* If we know well in advance as how many times a set of statements have to be repeatedly executed, then for loop is the best choice.

Syntax :-

```
for(exp1; exp2; exp3)
{
  statements;
}
```

where exp1 is initialization of variable
exp2 is condition
exp3 is increment / decrement.



1) WAP to print N Numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n, i;
  clrscr();
  printf("enter n\n");
  scanf("%d", &n);
  for(i=1; i<=n; i++)
  {
    printf("%d\n", i);
  }
  getch();
}
```

2) WAP to find factorial of given number using for loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i, fact=1, n;
 clrscr();
 printf("enter n\n");
 scanf("%d", &n);
 for(i=1; i<=n; i++)
 {
  fact = fact * i;
 }
 printf("factorial=%d", fact);
 getch();
}
```

✓ 3) WAP to generate n fibonacci series.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int i, n, fib1=0, fib2=1, fib3;
 clrscr();
 printf("enter n\n");
 scanf("%d", &n);
 if(n<2)
 {
  printf("fib series doesn't exist");
 }
```

```
else
{
printf("%d%d", fib1, fib2);
for(i=3; i<=n; i++)
{
fib3 = fib1+fib2;
printf("%d", fib3);
fib1=fib2;
fib2=fib3;
}
}
getch();
}
```


4) WAP to find whether a number is prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n, i, count=0;
printf("Enter n\n");
scanf("%d", &n);
for(i=1; i<=n; i++)
{
if(n%i==0)
{
count++;
}
}
if(count==2)
{
```

```
    printf("n is prime number\n");
}
else
{
printf("n is not prime number");
}
getch();
}
```

✓ 5) WAP to add even numbers & odd numbers between range 1 to n.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,i, esum, osum;
  clrscr();
  printf("Enter n\n");
  scanf("%d",&n);
  esum=0;
  osum=0;
  for(i=1; i<=n;i++)
  {
    if(i%2==0)
      esum=esum+i;
      else
      osum=osum+i;
  }

printf("sum of even numbers=%d\n", esum);
printf("sum of odd numbers=%d\n", osum);
getch();
}
```

6) WAP to add numbers which are divisible by 5 in a given range.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n1, n2, i, sum;
 clrscr();
 printf("Enter range n1 & n2\n");
 scanf("%d%d", &n1, &n2);
 sum=0;
 for(i=n1; i<=n2; i++)
 {
  if(i%5==0)
  {
   printf("%d\n", i);
   sum=sum+i;
  }
 }
 printf("sum=%d", sum);
 getch();
}
```

7) WAP to add series $1^2+2^2+3^2+\cdots+N^2$

Logic:-

```
int n, i, sum=0;
n=5;
for(i=1; i<=n; i++)
{
 sum=sum+pow(i,2);
}
printf("Sum=%d", sum);
```

Lab program 3:-

✓ Design & develop a C program to find the reverse of an integer number Num & check whether is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n, rev=0, digit=0, temp; clrscr();
  printf("Enter an integer number\n");
  scanf("%d", &n);
  temp=n;
  while(n!=0)
  {
    digit = n%10;
    rev=rev*10+digit;
    n=n/10;
  }
  printf("Given number is %d\n", temp);
  printf("Reverse=%d\n", rev);
  if(rev==temp)
  printf("Number is palindrome\n");
  else
  printf("Number is not palindrome\n");
  getch();
}
```

26/03/2017

prof. chaithra H.E

Module 4 :-

Arrays (part 1)

* Introduction
* Declaratn & initializatn of 1-D array
* Reading & printing of 1-D array
* Bubble sort
* Linear & Binary search
* Reading & printing of 2-D array
* programs on matrix operations: Addition, subtraction, multiplication, transpose

Introduction :-

```
int marks = 20;
```

In the above declaration, variables marks stores only one student marks.

Suppose if we want to store marks of 100 students, then we cant declare 100 variables. Instead of that we can use array.

we use arrays to store set of data items of same type.

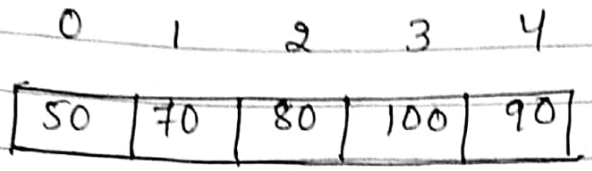
```
int marks[90];
```

In this, 90 fields will be allocated to store marks.

* Array :- Array is an ordered set of similar data items.

eg:- int marks[5];

Here, 5 fields will be allocated.



marks[0] = 50 → marks of 1st student
marks[4] = 90 → marks of 5th student

i.e., value can be accessed by using array index.

array index starts with 0 (zero).

In this example marks of 5 students are accessed by specifying marks[0] till marks[n-1] = marks[4].

Types of arrays

1) single dimensional array (1-D array)
2) Multi dimensional array (2-D, 3-D)

1) Single-dimensional array (1-D array)

It is a linear list consists of similar data items.

In memory data items are stored contiguos one after the other.

Array size depends on the data type we store

```
int a[5];
```

In this, array is 10 bytes.

i.e., Array size = n * sizeof(datatype)
= 5 * sizeof(int);
= 5 * 2
= 10 bytes

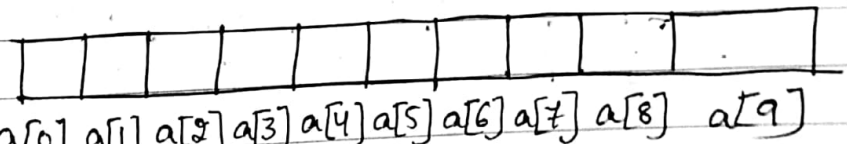
** Declaration of 1-D array

Syntax :- data_type array_name[size];
(size: positive integer)

Eg:- int a[10];

10 spaces will be reserved for 10 locations.

```
int a[10];
```



** Initialization of 1-D array

* Assigning values to an array.

Syntax :- data_type array_name[size] = {v1, ---, vn};
(v1, ---, vn: variable values.)

* data types can be int, float, char etc.,
* array_name is name of array
* size or expression should be always gives positive integer.

** Different ways of initializing arrays
** Compile time initialization :-

1) Initializing all specified memory location

Arrays can be initialized at the time of declaration when their initial values are known in advance.

eg:- int a[5] = { 10, 15, 20, 25, 30 };

During compilation, 5 contigues memory locations are reserved by the compiler for variable a & all these locations are initialized as shown below :-

a[0]	a[1]	a[2]	a[3]	a[4]
10	15	20	25	30

For characters,

int b[5] = { 'H', 'E', 'L', 'L', 'O' };

b[0]	b[1]	b[2]	b[3]	b[4]
H	E	L	L	O

2) partial array initialization.

Here, number of values initialized is less than the size of array.

Elements are initialized in the order from 0th location. The remaining locations will be initialized to zero automatically.

Eg:- int a[5] = {10, 20};

5 memory locations will be allocated & compiler initializes first 2 location with 10 & 20. The remaining memory locations are automatically initialized to 0's.

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	0	0	0

3) Initialization without size

Consider,

char b[] = {'H', 'E', 'L', 'L', 'O'};

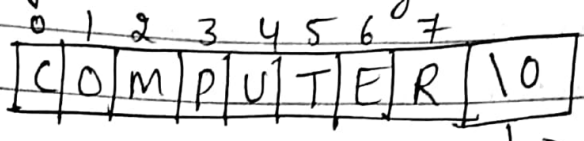
In this example, the array size is not specified. But array size will be set to total number of initial values specified. i.e., array size is automatically set to 5.

b[0]	b[1]	b[2]	b[3]	b[4]
H	E	L	L	O

4) Array initialization with a string

consider, char b[] = "COMPUTER";

Array b is initialized as below,



String "COMPUTER" contains 8 characters, but string always ends with null characters. So array size is 9 bytes (string length + 1).

** Run time initialization (Reading array)

* Reading & printing one-D array

consider, int a[5];

* 5 memory locations are allocated.
* Each elements can be accessed by specifying index.

* using a[0] through a[4], we can access 5 data items.

* The array can be read by using scanf() as follows:

```
scanf("%d", &a[0]);
:
:
scanf("%d", &a[n-1]);
```

In general, scanf("%d", &a[i]) where $i = 0, 1, \ldots, n-1$.
We can read n data items from keyboard as follows.

```
for(i=0; i<n; i++)
{
    scanf("%d", &a[i]);
}
```

Similarly, the n data items can be displayed using printf() as follows:

```
for(i=0; i<n; i++)
{
    printf("%d", a[i]);
}
```


WAP to read n elements & display the same using array.

```
void main()
{
    int n, a[10], i;
    printf("Enter no. of elements \n");
    scanf("%d", &n);
    printf("Enter n elements \n");
    for(i=0; i<n; i++)
    scanf("%d", &a[i]);
```

```
printf("The N elements are \n");
for(i=0; i<n; i++)
printf("%d\t", a[i]);
}
```

WAP to read n elements, find sum & display result using array.

```
void main()
{
  int n, i, a[10], sum=0;
  printf("Enter number of elements \n");
  scanf("%d", &n);
  printf("Enter N elements \n");
  for(i=0; i<n; i++)
  scanf("%d", &a[i]);
  for(i=0; i<n; i++)
  sum=sum+a[i];
  printf("sum=%d", sum);
  getch();
}
```

WAP to find sum of positive numbers, negative number using array.

```
void main()
{
  int n, i, a[10], psum=0, nsum=0;
  clrscr();
```

```
printf("Enter n\n");
scanf("%d", &n);
printf("Enter elements\n");
for(i=0; i<n; i++)
{
  scanf("%d", &a[i]);
}
for(i=0; i<n; i++)
{
  if(a[i]>=0)
    psum = psum+a[i];
  else
    nsum = nsum+a[i];
}
printf("Sum of +ve nos = %d\n", psum);
printf("Sum of -ve nos = %d\n", nsum);
getch();
}
```

CHAITHRA. H.E.
B.E, M. Tech

WAP to input n integer nos & conduct linear search for a given array elements.

```
void main()
{
  int n, i, a[100], key, found=0;
  clrscr();
  printf("Enter number of elements\n");
  scanf("%d", &n);
```

```
printf("Enter elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter key element to search\n");
scanf("%d",&key);
for(i=0;i<n;i++)
{
  if(a[i]==key)
  {
    found=1;
    break;
  }
}
if(found==1)
printf("key found at position=%d",i+1);
else
printf("not found");
getch();
}
```

CHAITHRA. H.E.
B.E, M. Tech