CONTROL STATEMENTS

When we run a program,the statements are executed in the order in which they appear in the program.Also each statement is executed only once.But in many cases we may need a statement or a set of statements to be executed a fixed no of times or until a condition is satisfied.Also we may want to skip some statements based on testing a condition.For all these we use control statements .
Control statements are of two types – branching and looping.

BRANCHING

It is to execute one of several possible options depending on the outcome of a logical test ,which is carried at some particular point within a program.

LOOPING

It is to execute a group of instructions repeatedly,a fixed no of times or until a specified condition is satisfied.

BRANCHING

1. **if else statement**

It is used to carry out one of the two possible actions depending on the outcome of a logical test.The else portion is optional.
The syntax is

If (expression) statement1 [if there is no else part]

Or

If (expression)

```
            Statement 1
        else
             Statement 2
```

Here expression is a logical expression enclosed in parenthesis.if expression is true ,statement 1 or statement 2 is a group of statements ,they are written as a block using the braces { }

Example:

```
1. if(x<0) printf("\n x is negative");
      2. if(x<0)
            printf("\n x is negative");
                else
     printf("\n x is non negative");

       3.if(x<0)
                 {
                  x=-x;
      s=sqrt(x);
      }
         else
                 s=sqrt(x);
```

2. nested if statement

Within an if block or else block another if – else statement can come. Such statements are called nested if statements.

The syntax is

```
If (e1)
   s1
     if (e2)
         s2
       else
         s3
        else
```

3. Ladder if statement

Inorder to create a situation in which one of several courses of action is executed we use ladder – if statements.

The syntax is

```
If (e1) s1
 else if (e2) s2
    else if (e3) s3
         ……………….
              else sn
```

Example:
```
if(mark>=90) printf("\n excellent");
   else if(mark>=80) printf("\n very good");
      else if(mark>=70) printf("\n good");
         else if(mark>=60) printf("\n average");
             else
                printf("\n to be improved");
```

SWITCH STATEMENT

It is used to execute a particular group of statements to be chosen from several available options. The selection is based on the current value of an expression with the switch statement.

The syntax is:

```
switch(expression)
{
   case value1:
        s1
        break;
  case value 2:
         s2
         break;
         …….
         ……..
      default:
          sn
}
```

All the option are embedded in the two braces { }.Within the block each group is written after the label case followed by the value of the expression and a colon. Each group ends with '***break'*** statement. The last may be labeled ***'default'.*** This is to avoid error and to execute the group of statements in default if the value of the expression does not match value1, value2,……..

LOOPING

1. The while statement

This is to carry out a set of statements to be executed repeatedly until some condition is satisfied.

The syntax is:

While (expression) statement

The statement is executed so long as the expression is true. Statement can be simple or compound.

Example 1:

```
#include<stdio.h>
while(n > 0)
{
printf("\n");
n = n - 1;
}
```

Example 2:

```
#include<stdio.h>
            main()
           {
                  int i=1;
                 while(x<=10)
                       {
                        printf("%d",i);
                        ++i;
                        }
               }
```

2. do while statement

This is also to carry out a set of statements to be executed repeatedly so long as a condition is true.

The syntax is:

do statement while(expression)

Example:

```
#include<stdio.h>
        main()
       {
           int i=1;
            do
           {
                  printf("%d",i);
                  ++i;
           }while(i<=10);




}
```

THE DIFFERENCE BETWEEN while loop AND do – while loop

1) In the while loop the condition is tested in the beginning whereas in the other case it is done at the end.
2) In while loop the statements in the loop are executed only if the condition is true.whereas in do – while loop even if the condition is not true the statements are executed atleast once.

3. for loop

It is the most commonly used looping statement in C. The general form is

For(expression1;expression2;expression3)statement

Here expression1 is to initialize some parameter that controls the looping action.expression2 is a condition and it must be true to carry out the action.expression3 is a unary expression or an assignment expression.

Example:

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

Here the program prints *i* starting from 1 to 10.First *i* is assigned the value 1 and than it checks whether *i*<=10 If so i is printed and then *i* is increased by one. It continues until *i*<=10.

An example for finding the average of 10 numbers;

```
#include<stdio.h>
main()
{
  int i;
  float x,avg=0;
  for(i=1;i<=10;++i)
      {
        scanf("%f",&x);
        avg += x;
       }
        avg /= 10;
         printf("\n average=%f",avg);
 }
```

Note: Within a loop another for loop can come

Example :

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

The break statement

The break statement is used to terminate4 loop or to exit from a switch. It is used in for, while, do-while and switch statement.

The syntax is ***break;***

Example 1: A program to read the sum of positive numbers only

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

```
main()
 {
     int x, sum=0;
      int n=1;
     while(n<=10)
         {
              scanf(“%d”,&x);
    if(x<0) break;
                  sum+=x;
          }
     printf(“%d”,sum);
  }
```

Example 2 :A program for printing prime numbers between 1 and 100:

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

main()
{
int i, j;

printf("%d\n", 2);

for(i = 3; i <= 100; i = i + 1)
        {
        for(j = 2; j < i; j = j + 1)
                {
                if(i % j == 0)
                        break;
                if(j > sqrt(i))
                        {
                        printf("%d\n", i);
                        break;
                        }
                }
        }

return 0;
}
```

Here while loop breaks if the input for x is –ve.

The continue statement

It is used to bypass the remainder of the current pass through a loop. The loop does not terminate when continue statement is encountered, but statements after continue are skipped and proceeds to the next pass through the loop.

In the above example of summing up the non negative numbers when a negative value is input, it breaks and the execution of the loop ends. In case if we want to sum 10 nonnegative numbers, we can use *continue* instead of *break*

Example :
```
#include<stdio.h>
       main()
       {
            int x, sum=0, n=0;
             while(n<10)
             {
                  scanf("%d",x);
                  if(x<0) continue;
                         sum+=x;
                        ++n;
            }
                  printf("%d",sum);
      }
```

<u>GO TO statement</u>

It is used to alter the normal sequence of program execution by transferring control to some other part of the program .The syntax is goto *label* ;

Example :

```
 #include<stdio.h>
main( )
{
   int n=1,x,sum=0;
   while(n<=10)
   {
      scanf("%d" ,&x);
      if(x<0)goto error;
      sum+=x;
      ++n;
    }
 error:
  printf("\n the number is non negative");
}
```