2.6 Assignment Statement

Assignment statement can be defined as the statement through which the value obtained from an expression can be stored in a variable.

The general form of assignment statement is

```
< variable name> = < arithmetic expression> ;
```

```
Example:  sum  =  a + b + c;
                tot  =  s1 + s2 + s3;
                area  =  ½ * b* h;
```

2.7 I/O Control Structure (if, If-else, for, while, do-while)

Conditional Statements

The conditional expressions are mainly used for decision making. The following statements are used to perform the task of the conditional operations.

a. if statement.

b. If-else statement. Or 2 way if statement

c. Nested else-if statement.

d. Nested if–else statement.

e. Switch statement.

a. if statement

The **if statement** is used to express conditional expressions. If the given condition is true then it will execute the statements otherwise skip the statements.

The simple structure of '**if**' statement is

```
i.      If (< condtional expressione>)
              statement-1;
              (or)
ii.     If (< condtional expressione>)
              {
```

```
statement-1;
statement-2;
statement-3;
……………
……………
STATEMENT-N
}
```

The expression is evaluated and if the expression is true the statements will be executed. If the expression is false the statements are skipped and execution continues with the next statements.

Example: a=20; b=10;

```
if ( a > b )
printf (“big number is %d” a);
```

b. if-else statements

The **if-else** statements is used to execute the either of the two statements depending upon the value of the exp. The general form is

```
if(<exp>)
{
Statement-1;
Statement -2;
…………..          “ SET-I”
……………
Statement- n;
}
else
{
Statement1;
Statement 2;
```

```
        ………….. " SET-II
        ……………
        Statement n;
        }
```

SET - I Statements will be executed if the exp is true.

SET – II Statements will be executed if the exp is false.

Example:

```
        if ( a> b )
        printf ("a is greater than b");
        else
        printf ("a is not greater than b");
```

c. Nested else-if statements

If some situations if may be desired to nest multiple **if-else** statements. In this situation one of several different course of action will be selected.

Syntax

```
if ( <exp1> )
        Statement-1;
else if ( <exp2> )
        Statement-2;
else if ( <exp3> )
        Statement-3;
else
        Statement-4;
```

When a logical expression is encountered whose value is true the corresponding statements will be executed and the remainder of the nested else if statement will be bypassed. Thus control will be transferred out of the entire nest once a true condition is encountered.

The final **else** clause will be apply if none of the exp is true.

d. nestedif-else statement

It is possible to nest if-else statements, one within another. There are several different form that nested if-else statements can take.

The most general form of two-layer nesting is

```
        if(exp1)
                if(exp3)
            Statement-3;
                else
            Statement-4;
        else
                if(exp2)
            Statement-1;
                else
            Statement-2;
```

One complete **if-else** statement will be executed if **expression1** is true and another complete **if-else** statement will be executed if **expression1** is false.

e. Switch statement

A switch statement is used to choose a statement (for a group of statement) among several alternatives. The switch statements is useful when a variable is to be compared with different constants and in case it is equal to a constant a set of statements are to be executed.

Syntax:

```
        Switch (exp)
        {
        case
            constant-1:
            statements1;
        case
            constant-2:
```

```
        statements2;
        _______
        _______
    default:
        statement n;
    }
```

Where constant1, constanat2 — — — are either integer constants or character constants. When the switch statement is executed the exp is evaluated and control is transferred directly to the group of statement whose case label value matches the value of the exp. If none of the case label values matches to the value of the exp then the default part statements will be executed.

If none of the case labels matches to the value of the exp and the default group is not present then no action will be taken by the switch statement and control will be transferred out of the switch statement.

A simple switch statement is illustrated below.

Example 1:

```
main()
{
char choice;
printf("Enter Your Color (Red - R/r, White – W/w)");
choice=getchar();
switch(choice= getchar())
{
case'r':
case'R':
printf("Red");
break;
case'w':
case'W':
```

```
printf("white");
break;
default :
printf("no colour");
}
```

Example 2:

```
switch(day)
{
case 1:
printf("Monday");
break;
______
______
}
```

2.8 Structure for Looping Statements

Loop statements are used to execute the statements repeatedly as long as an expression is true. When the expression becomes false then the control transferred out of the loop. There are three kinds of loops in **C**.

a) while b) do-while c) for

a. while statement

while loop will be executed as long as the exp is true.

Syntax:

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

The statements will be executed repeatedly as long as the exp is true. If the exp is false then the control is transferred out of the while loop.

Example: