Unit –II

Unit-2	
Statement	A statement is a syntactic unit of an imperative programming language that expresses some action to be carried out
Iteration Statement	Iteration statements cause statements (or compound statements) to be executed zero or more times, subject to some loop-termination criteria.
Array	Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.
Block	The block collects statements together into a single compound statements.
String	Strings are actually one-dimensional array of characters terminated by a null character '\0'.

Concepts:

Control statements in C Programming Language:

To control the flow of program execution c programming languages uses control statements.

C provides two sytles of flow control:

- Branching (or) Selectional statements
- Looping (or) Iterative statements

Branching is deciding what actions to take and looping is deciding how many times to take a certain action.

Selectional statement are used to make one-time decisions in C Programming, that is, to execute some code/s and ignore some code/s depending upon the test expression.

IF STATEMENT

```
if (test expression)
{
        statement/s to be executed if test expression is true;
}
```

The if statement checks whether the text expression inside parenthesis () is true or not. If the test expression is true, statement/s inside the body of if statement is executed but if test is false, statement/s inside body of if is ignored.

Flowchart of if statement

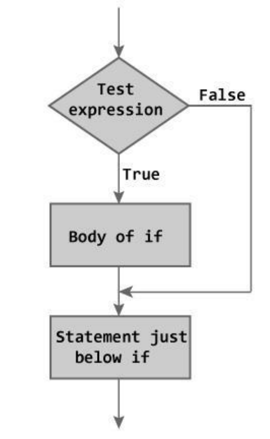


Figure: Flowchart of if Statement

C IF...ELSE STATEMENT

The if...else statement is used if the programmer wants to execute some statement/s when the test expression is true and execute some other statement/s if the test expression is false.

Syntax of if...else

```
if (test expression) {
statements to be executed if test expression is true;
}
else {
statements to be executed if test expression is false;
}
```

Flowchart of if...else statement

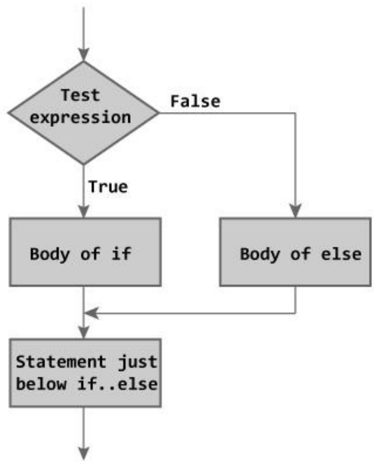


Figure: Flowchart of if...else Statement

Nested if...else statement (if...elseif....else Statement)

The nested if...else statement is used when program requires more than one test expression.

Syntax of nested if...else statement.

```
if (test expression1){
statement/s to be executed if test expression1 is true;
}
else if(test expression2) {
statement/s to be executed if test expression1 is false and 2 is true;
}
else if (test expression 3) {
statement/s to be executed if text expression1 and 2 are false and 3 is true;
}
.
.
.
else {
statements to be executed if all test expressions are false;
}
```

How else if ladder.. works?

The nested if...else statement has more than one test expression. If the first test expression is true, it executes the code inside the braces{ } just below it. But if the first test expression is false, it checks the second test expression. If the second test expression true, it executes the statement/s inside the braces{ } just below it. This process continues. If all the test expression are false, code/s inside else is executed and the control of program jumps below the nested if...else
The ANSI standard specifies that 15 levels of nesting may be continued.

C PROGRAMMING LOOPS

Loops cause program to execute the certain block of code repeatedly until test condition is false. Loops are used in performing repetitive task in programming. Consider these scenarios:

1. You want to execute some code/s 100 times.
2. You want to execute some code/s certain number of times depending upon input from user.

These types of task can be solved in programming using loops.

There are 3 types of loops in C programming:

1. for loop
2. while loop
3. do...while loop

for Loop Syntax

```
for(initialization statement; test expression; update statement) {
code/s to be executed;
}
```

How for loop works in C programming?

The initialization statement is executed only once at the beginning of the for loop. Then the test expression is checked by the program. If the test expression is false, for loop is terminated. But if test expression is true then the code/s inside body of for loop is executed and then update expression is updated. This process repeats until test expression is false.

This flowchart describes the working of for loop in C programming.

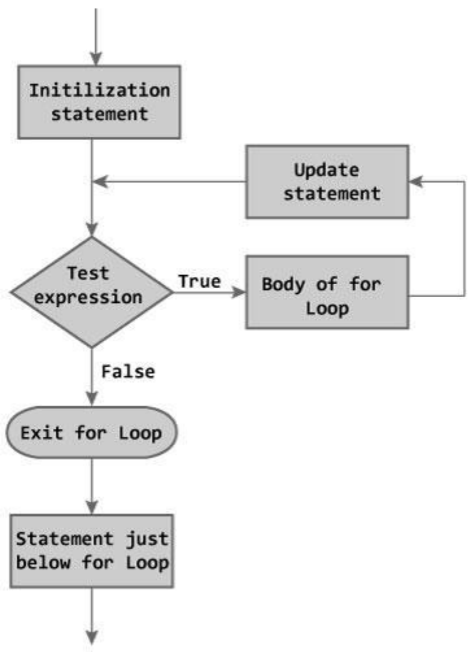


Figure: Flowchart of for Loop

WHILE LOOP STATEMENT:

Syntax of while loop

```
while (test expression) {
statement/s to be executed.
}
```

The while loop checks whether the test expression is true or not. If it is true, code/s inside the body of while loop is executed,that is, code/s inside the braces { } are executed. Then again the test expression is checked whether test expression is true or not. This process continues until the test expression becomes false.

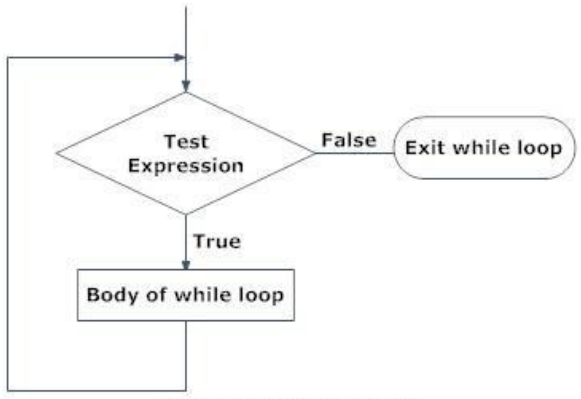


Figure: Flowchart of while loop

Do...while loop

In C, do...while loop is very similar to while loop. Only difference between these two loops is that, in while loops, test expression is checked at first but, in do...while loop code is executed at first then the condition is checked. So, the code are executed at least once in do...while loops.

Syntax of do...while loops

```
do {
some code/s;
}
while (test expression);
```

At first codes inside body of do is executed. Then, the test expression is checked. If it is true, code/s inside body of do are executed again and the process continues until test expression becomes false(zero).

Notice, there is semicolon in the end of while (); in do...while loop.

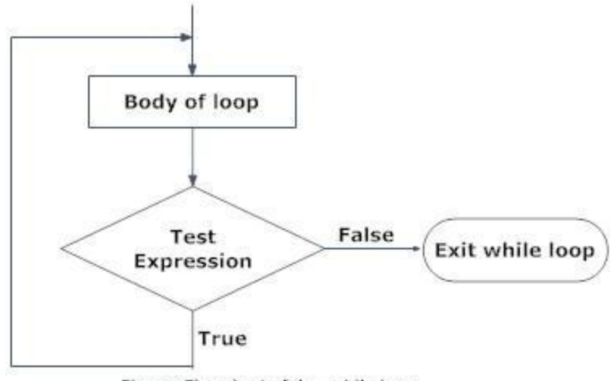


Figure: Flowchart of do...while loop

BREAK & CONTINUE STATEMENT:

There are two statements built in C programming, break; and continue; to alter the normal flow of a program. Loops perform a set of repetitive task until text expression becomes false but it is sometimes desirable to skip some statement/s inside loop or terminate the loop immediately without checking the test expression. In such cases, break and continue statements are used. The break;statement is also used in switch statement to exit switch statement.

break Statement

In C programming, break is used in terminating the loop immediately after it is encountered. The break statement is used with conditional if statement.

Syntax of break statement

break;

The break statement can be used in terminating all three loops for, while and do...while loops.

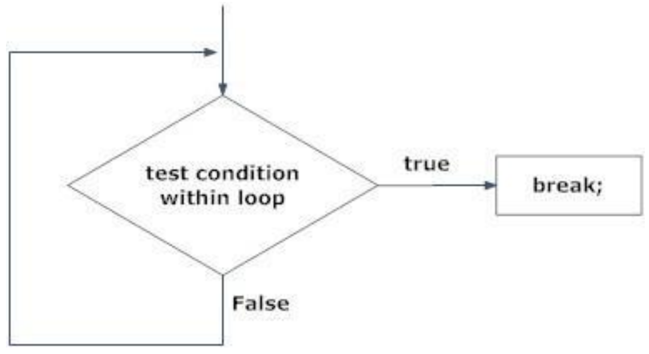


Figure: Flowchart of break statement

The figure below explains the working of break statement in all three type of loops.

CONTINUE STATEMENT

It is sometimes desirable to skip some statements inside the loop. In such cases, continue statements are used.

Syntax of continue Statement

continue;

Just like break, continue is also used with conditional if statement.

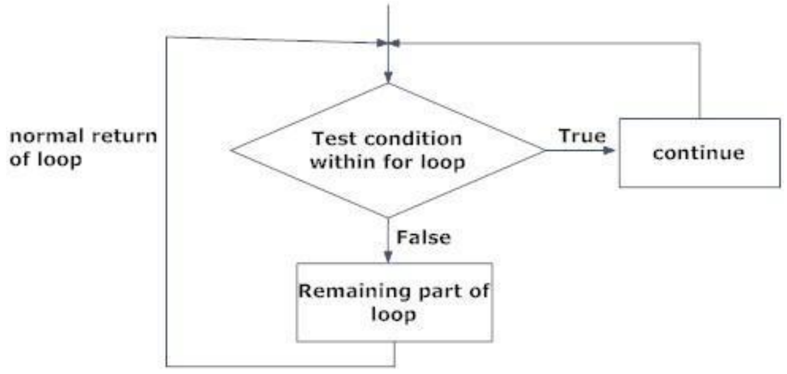


Fig: Flowchart of continue statement

SWITCH CASE STATEMENT:

Decision making are needed when, the program encounters the situation to choose a particular statement among many statements. If a programmer has to choose one block of statement among many alternatives, nested if...else can be used but, this makes programming logic complex. This type of problem can be handled in C programming using switch statement.

Syntax of switch...case

switch (*n*) {

case constant1:

code/s to be executed if *n* equals to *constant1*;

break;

case *constant2*:

code/s to be executed if *n* equals to *constant2*;

break;

.

.

.

default:

code/s to be executed if *n* doesn't match to any cases;

}

The value of *n* is either an integer or a character in above syntax. If the value of *n* matches constant in case, the relevant codes are executed and control moves out of the switch statement. If the *n*doesn't matches any of the constant in case, then the default codes are executed and control moves out of switch statement.

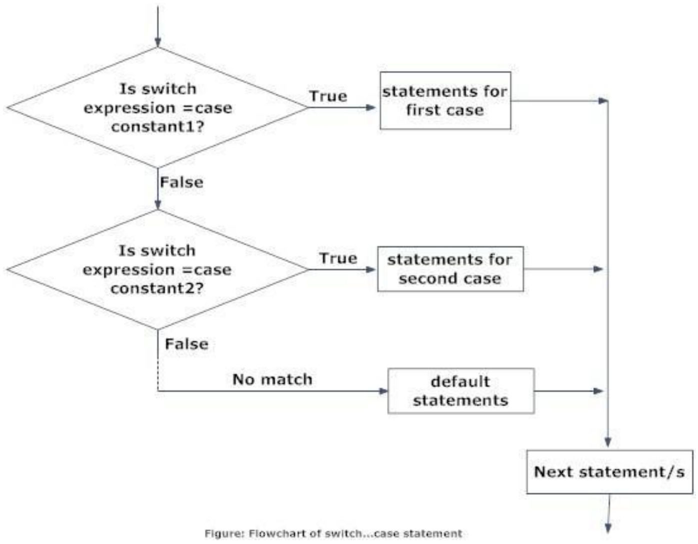

Is switch expression =case constant1?
True
statements for first case
False
Is switch expression =case constant2?
True
statements for second case
False
No match
default statements
Next statement/s

Figure: Flowchart of switch...case statement

GOTO STATEMENT:

In C programming, goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

Syntax of goto statement

goto label;

....

....

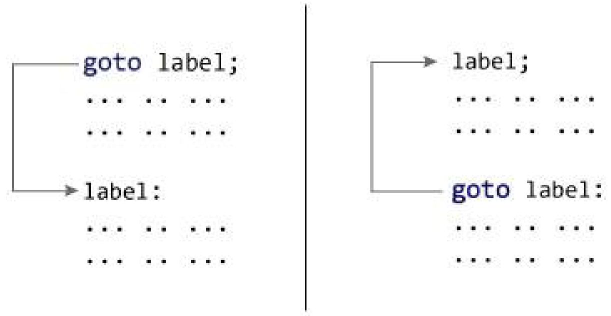
....

label:

statement;

In this syntax, label is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the label: and executes the code below it.

Example:



Reasons to avoid goto statement

Though, using goto statement give power to jump to any part of program, using goto statement makes the logic of the program complex and tangled. In modern programming, goto statement is considered a harmful construct and a bad programming practice.

The goto statement can be replaced in most of C program with the use of break and continue statements. In fact, any program in C programming can be perfectly written without the use of goto statement. All programmer should try to avoid goto statement as possible as they can.

C PROGRAMMING ARRAYS

In C programming, one of the frequently arising problem is to handle similar types of data. For example: If the user want to store marks of 100 students. This can be done by creating 100 variable individually but, this process is rather tedious and impracticable. These type of problem can be handled in C programming using arrays.

An array is a sequence of data item of homogeneous value(same type).

Arrays are of two types:

1. One-dimensional arrays
2. Multidimensional arrays

Declaration of one-dimensional array

data_type array_name[array_size];

For example:

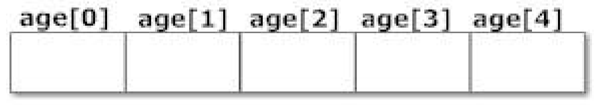
int age[5];

Here, the name of array is *age*. The size of array is 5,i.e., there are 5 items(elements) of array *age*. All element in an array are of the same type (int, in this case).

Array elements

Size of array defines the number of elements in an array. Each element of array can be accessed and used by user according to the need of program. For example:

int age[5];



Array elements

Note that, the first element is numbered 0 and so on.

Here, the size of array *age* is 5 times the size of int because there are 5 elements.

Suppose, the starting address of age[0] is 2120d and the size of int be 4 bytes. Then, the next address (address of a[1]) will be 2124d, address of a[2] will be 2128d and so on.

Initialization of one-dimensional array:

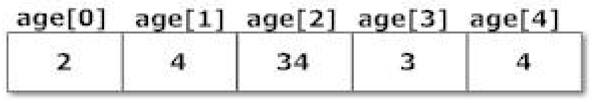
Arrays can be initialized at declaration time in this source code as:

int age[5]={2,4,34,3,4};

It is not necessary to define the size of arrays during initialization.

int age[]={2,4,34,3,4};

In this case, the compiler determines the size of array by calculating the number of elements of an array.



age[0]	age[1]	age[2]	age[3]	age[4]
2	4	34	3	4

Initialization of one-dimensional array

Accessing array elements

In C programming, arrays can be accessed and treated like variables in C.

For example:

```
scanf("%d",&age[2]);
/* statement to insert value in the third element of array age[]. */
scanf("%d",&age[i]);
/* Statement to insert value in (i+1)th element of array age[]. */
/* Because, the first element of array is age[0], second is age[1], ith is age[i-1] and (i+1)th is age[i]. */
printf("%d",age[0]);
/* statement to print first element of an array. */
printf("%d",age[i]);
/* statement to print (i+1)th element of an array. */
```

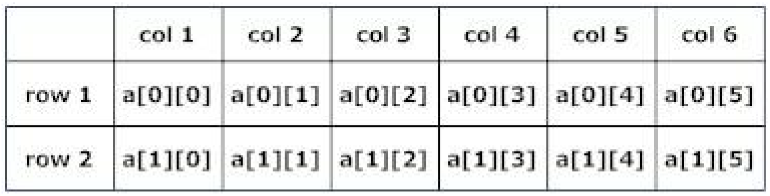
Multidimentional arrays:

C programming language allows programmer to create arrays of arrays known as multidimensional arrays. For example:

```
float a[2][6];
```

Here, *a* is an array of two dimension, which is an example of multidimensional array.

For better understanding of multidimensional arrays, array elements of above example can be thinked of as below:



	col 1	col 2	col 3	col 4	col 5	col 6
row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]

Figure: Multidimensional Arrays

Initialization of Multidimensional Arrays

In C, multidimensional arrays can be initialized in different number of ways.

```
int c[2][3]={{1,3,0}, {-1,5,9}};
```

OR

```
int c[][3]={{1,3,0}, {-1,5,9}};
```

OR

int c[2][3]={1,3,0,-1,5,9};

Initialization Of three-dimensional Array

double cprogram[3][2][4]={

{{-0.1, 0.22, 0.3, 4.3}, {2.3, 4.7, -0.9, 2}},

{{0.9, 3.6, 4.5, 4}, {1.2, 2.4, 0.22, -1}},

{{8.2, 3.12, 34.2, 0.1}, {2.1, 3.2, 4.3, -2.0}}

};

Suppose there is a multidimensional array arr[i][j][k][m]. Then this array can hold i*j*k*m numbers of data.

Similarly, the array of any dimension can be initialized in C programming.

Assignment Questions

Unit -II

1. Differentiate between elseif and switch statements with examples.
2. Give a note on iteration statements in C language.
3. With Suitable C programs, describe Two Way Selection and Multiway Selection. Explain the differences between them.
4. What is an array? What are advantages of arrays over ordinary variables? How arrays are declared and initialized?
5. Give the points of similarity and differences between a while loop and a do while loop.
6. Explain the differences between single dimensional & multi dimensional arrays in C?
7. Explain the ways in which we can pass a one dimension array to functions.
8. Distinguish between the break and the continue statement.