**Unit 4 :**

**Control Statement**

Generally C program statement is executed in a order in which they appearin the program. But sometimes we use decision making condition for execution only a part of program, that is called control statement. Control statement defined how the control is transferred from one part to the other part of the program.

There are several control statement like if...else, switch, while, do....while, for loop, break, continue, goto etc.

**Loops in C**

***Loop:-***it is a block of statement that performs set of instructions. In loops

Repeating particular portion of the program either a specified number of time or

until a particular no of condition is being satisfied.

**There are three types of loops in c**

1.While loop

2.do while loop

3.for loop

**While loop**

**Syntax :**

while(condition)

{

Statement 1;

Statement 2;

}

Or while(test condition)

Statement;

**do while loop**

This (do while loop) statement is also used for looping. The body of this loop may contain single statement or block of statement. The syntax for writing this statement is:

**Syntax:-**

Do

{

Statement;

}

while(condition);

**for loop**

In a program, for loop is generally used when number of iteration are known in advance. The body of the loop can be single statement or multiple statements. Its syntax for writing is:

**Syntax:**

for(exp1;exp2;exp3)

{

Statement;

}

Or

for(initialized counter; test counter; update counter)

{

Statement;

}

Here exp1 is an initialization expression, exp2 is test expression or condition and exp3 is an update expression. Expression 1 is executed only once when loop started and used to initialize the loop variables. Condition expression generally uses relational and logical operators. And updation part executed only when after body of the loop is executed.

**Nesting of loop**

When a loop written inside the body of another loop then, it is known as nesting of loop. Any type of loop can be nested in any type such as while, do while, for. For example nesting of for loop can be represented as :

void main()

{

int i,j;

for(i=0;i<2;i++)

for(j=0;j<5; j++)

printf(“%d %d”, i, j);

}

Output: i=0

j=0 1 2 3 4

i=1

j=0 1 2 3 4

**Break statement(break)**

* Sometimes it becomes necessary to come out of the loop even before loop condition becomes false then break statement is used.
* Break statement is used inside loop and switch statements.
* It cause immediate exit from that loop in which it appears and it is generally written with condition.
* It is written with the keyword as break.
* When break statement is encountered loop is terminated and control is transferred to the statement, immediately after loop or situation where we want to jump out of the loop instantly without waiting to get back to conditional state.
* When break is encountered inside any loop, control automatically passes to the first statement after the loop. This break statement is usually associated with if statement.

**Example :**

void main()

{

int j=0;

for(;j<6;j++)

if(j==4)

break;

}

**Continue statement (key word continue):**

* Continue statement is used for continuing next iteration of loop after skipping some statement of loop.
* When it encountered control automatically passes through the beginning of the loop.
* It is usually associated with the if statement. It is useful when we want to continue the program without executing any part of the program.
* The difference between break and continue is, when the break encountered loop is terminated and it transfer to the next statement and when continue is encounter control come back to the beginning position.
* In while and do while loop after continue statement control transfer to the test condition and then loop continue where as in, for loop after continue control transferred to the updating expression and condition is tested.

**if statement**

Statement execute set of command like when condition is true and its syntax is

If (condition)

Statement;

The statement is executed only when condition is true. If the if statement body is consists of several statement then better to use pair of curly braces. Here in case condition is false then compiler skip the line within the if block.

**if…..else ... Statement**

* It is bidirectional conditional control statement that contains one condition & two possible action. Condition may be true or false, where non-zero value regarded as true & zero value regarded as false.
* If condition are satisfy true, then a single or block of statement executed otherwise another single or block of statement is executed.

**Its syntax is:-**

if (condition)

{

Statement1;

Statement2;

}

else

{

Statement1;

Statement2;

}

Else statement cannot be used without if or no multiple else statement are allowed within one if statement. It means there must be a if statement with in an else statement

**Nesting of if …else**

When there are another if else statement in if-block or else-block, then it is called nesting of if-else statement.

**Syntax is :-**

if (condition)

{

If (condition)

Statement1;

else

statement2;

}

Statement3;

**If….else LADDER**

In this type of nesting there is an if else statement in every else part except the last part. If condition is false control pass to block where condition is again checked with its if statement.

Syntax is :-

if (condition)

Statement1;

else if (condition)

statement2;

else if (condition)

statement3;

else

statement4;

This process continue until there is no if statement in the last block. if one of the condition is satisfy the condition other nested “else if” would not executed. But it has disadvantage over if else statement that, in if else statement whenever the condition is true, other condition are not checked. While in this case, all condition are checked.

**ARRAY**

* Array is the collection of similar data types or collection of similar entity stored in contiguous memory location.
* Array of character is a string.
* Each data item of an array is called an element. And each element is unique and located in separated memory location.
* Each of elements of an array share a variable but each element having different index no. known as subscript.
* An array can be a single dimensional or multi-dimensional and number of subscripts determines its dimension.
* And number of subscript is always starts with zero.
* One dimensional array is known as vector and two dimensional arrays are known as matrix.

**ADVANTAGES:**

array variable can store more than one value at a time where other variable can store one value at a time.

Example:

int arr[100];

**DECLARATION OF AN ARRAY :**

Its syntax is :

Data type array name [size];

int arr[100];

int mark[100];

int a[5]={10,20,30,100,5}

* The declaration of an array tells the compiler that, the data type, name of the array, size of the array and for each element it occupies memory space.
* Like for int data type, it occupies 2 bytes for each element and for float it occupies 4 byte for each element etc.
* The size of the array operates the number of elements that can be stored in an array and it may be a int constant or constant int expression.

We can represent individual array as :

int ar[5];

ar[0], ar[1], ar[2], ar[3], ar[4];

Symbolic constant can also be used to specify the size of the array as:

#define SIZE 10;

**INITIALIZATION OF AN ARRAY:**

After declaration element of local array has garbage value. If it is global or static array then it will be automatically initialize with zero. An explicitly it can be initialize that

Data type array name [size] = {value1, value2, value3…}

**Write a program to add 10 array elements \*/**

#include<stdio.h>

void main()

{

int i ;

int arr [10];

int sum=o;

for (i=0; i<=9; i++)

{

printf (“enter the %d element \n”, i+1);

scanf (“%d”, &arr[i]);

}

for (i=0; i<=9; i++)

{

sum = sum + a[i];

}

printf (“the sum of 10 array elements is %d”, sum);

}

**Two dimensional arrays**

Two dimensional array is known as matrix. The array declaration in both the array i.e.in single dimensional array single subscript is used and in two dimensional array two subscripts are is used.

**Its syntax is**

Data-type array name[row][column];

Or we can say 2-d array is a collection of 1-D array placed one below the other.

Total no. of elements in 2-D array is calculated as row\*column

Example:-

int a[2][3];

Total no of elements=row\*column is 2\*3 =6

It means the matrix consist of 2 rows and 3 columns

**Initialization of 2-d array:**

2-D array can be initialized in a way similar to that of 1-D array.

for example:-int mat[4][3]={11,12,13,14,15,16,17,18,19,20,21,22};

These values are assigned to the elements row wise, so the values of elements after this initialization are

Mat[0][0]=11, Mat[1][0]=14, Mat[2][0]=17 Mat[3][0]=20

Mat[0][1]=12, Mat[1][1]=15, Mat[2][1]=18 Mat[3][1]=21

Mat[0][2]=13, Mat[1][2]=16, Mat[2][2]=19 Mat[3][2]=22

**String**

Array of character is called a string. It is always terminated by the NULL character. String is a one dimensional array of character.

We can initialize the string as

char name[]={‘j’,’o’,’h’,’n’,’\o’};

Here each character occupies 1 byte of memory and last character is always NULL character. Where ’\o’ and 0 (zero) are not same, where ASCII value of ‘\o’ is 0 and ASCII value of 0 is 48. Array elements of character array are also stored in contiguous memory allocation.

From the above we can represent as;

J o h N ‘\o’

The terminating NULL is important because it is only the way that the function that work with string can know, where string end.

String can also be initialized as;

char name[]=”John”;

Here the NULL character is not necessary and the compiler will assume it automatically.

**String library function**

There are several string library functions used to manipulate string and the prototypes for these functions are in header file “string.h”. Several string functions are

**strlen()**

This function return the length of the string. i.e. the number of characters in the string excluding the terminating NULL character. It accepts a single argument which is pointer to the first character of the string.

**strcmp()**

This function is used to compare two strings. If the two string match, strcmp() return a value 0 otherwise it return a non-zero value. It compare the strings character by character and the comparison stops when the end of the string is reached or the corresponding characters in the two string are not same.

**strcpy() :**

This function is used to copying one string to another string. The function strcpy(str1,str2) copies str2 to str1 including the NULL character. Here str2 is the source string and str1 is the destination string. The old content of the destination string str1 are lost. The function returns a pointer to destination string str1.

**strcat():** This function is used to append a copy of a string at the end of the other string. If the first string is “”Purva” and second string is “Belmont” then after using this function the string becomes “PusvaBelmont”.

The NULL character from str1 is moved and str2 is added at the end of str1. The 2nd string str2 remains un affected.

A pointer to the first string str1 is returned by the function.