Programming in ` C ` Language

INTRODUCTION

i. 'C' was an offspring of the B Language, developed in 1960.

ii. BCPL – Basic combined programming Language.,

iii. B was modified by Dennis Ritchie and was implemented at Bell Lab.

iv. It's strongly associated with UNIX.

Importance of 'C'

i. It's a robust Language whose rich set of built-in functions and operators can be used to write any complex program.

ii. 'c' compiler combines the assembly Language with high level Language.

iii. Very suited for both system software & business packages

iv. Efficient and fast. It's many times faster than BASIC (approximately 15000)

v. Highly portable

vi. Structured programming Language

vii. Program debugging, testing and maintains easier

viii. C program is collection of function.

ix. C is high level programming Language

Structure of 'C' programs

Documentation section

Link section

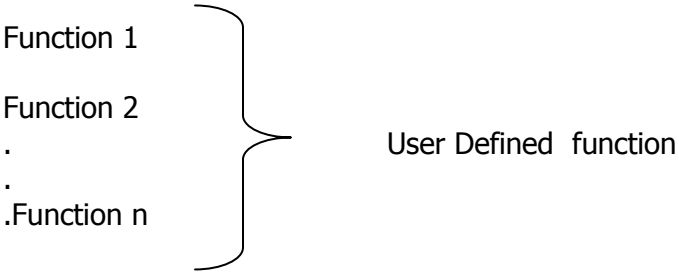
Definition section

Global declaration section

main() function section
{
Declaration part;

Executable part;
}

sub program section



The **Documentation section** consists of a set of comment Lines such as name of program, author and other details.

The **Link** section to the compiler to link functions from the system Library.

The **definition section** defines all symbolic constants.

The **global declaration** section declare variable in the outside of all the functions.
A variable used more than one functions is called global variable.

Every 'c' program must have one **main ( ) function**.
This section contains two parts,

i. Declaration part
ii. Executable part.

The Declaration part declares all the variables used in the executable part.

The program execution begins at the opening brace and ends at the closing brace.

All statements in the declaration and executable parts end with a semicolon.

The **subprogram section** contains al the user –defined functions are called in the main function.

Programming style

1. 'c' is a free form Language
2. Increase the readability.
3. Distinction between uppercase & lower case Letters

main( )

- It's a special function to start the program
- 'c' program must have one main() function.

Executing 'c' program:

1. Creating the program
2. Compiling the program
3. Linking the program (Link to Library)
4. Executing the program

The compiles and Linked program is called the executable object code and stores automatically filename out.

(ex) sample. out

Program

The task of processing of data is accomplished by executing a sequence of precise instruction called a program.

CHARACTER SET

1. Letters (A…Z)

2. Digits (0…9)

3. Special characters

4. White spaces (Blank space, Horizontal tab, carriage return, new line, form feed)

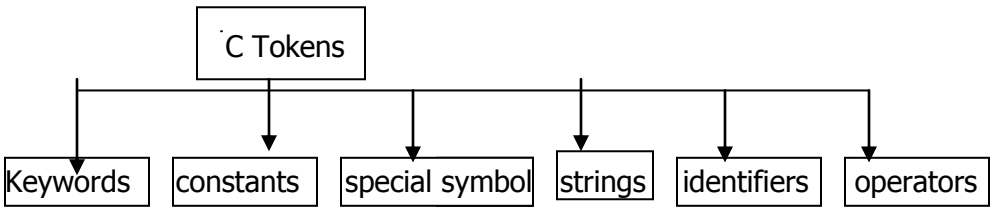
Special charactors:

` ,&, n, *, -, +, <, >,#, /, \, ~, $, etc..,

C TOKENS

Individual words and punctuation marks are called tokens.

Six types of tokens:



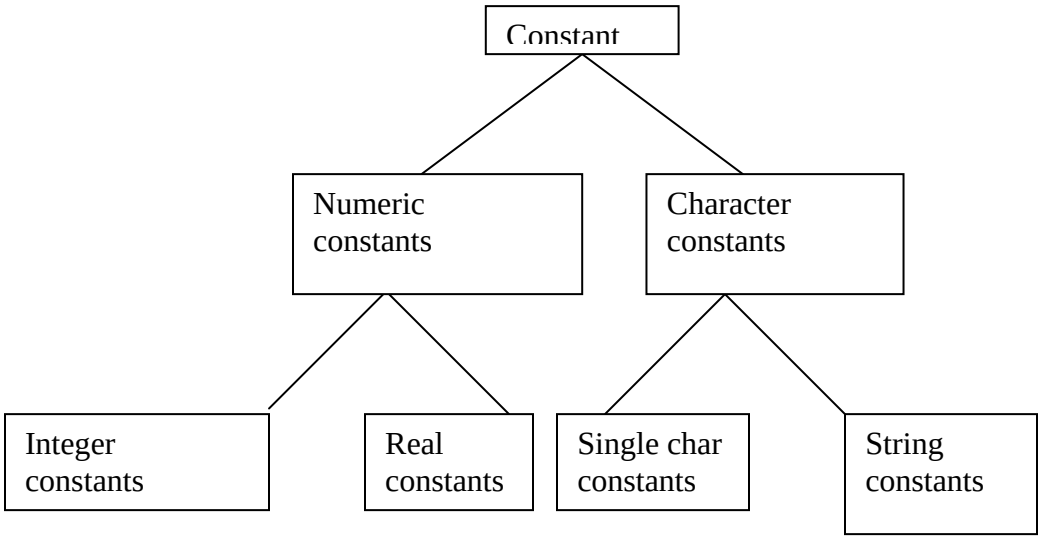
Key Words

- All keywords have fixed meanings and these meanings can't be changed.
- All keywords written in lowercase
- 'C' contains 32 keywords.

(ex) auto, break , int , float, chars etc.,

CONSTANTS:

Fixed values that doesn't change during the executing of a program.



Integer Constants

Refers to a sequence of digits.

There types,

i. Decimal
ii. Octal
iii. Hexa decimal

(Ex) 1 2 3 , - 1 2 1, o, +78, 6584641 etc
ox, oxbcd etc.,

- spaces, commas, non-digit characters are not permitted.

Real constants

The Fractional or the precision numbers are called real constant.

(ex) 215.12, 0.001 etc

Real number may also expressed in exponential form.

Mantissa e exponent

- The mantissa is either a real number expressed in fractional number.
- The exponent is an integer number with optional plus or minus sign.
- The Letter e separating mantissa and the exponent

Single char const:

It contains a single character enclosed within a pair of single Quote marks.

(Ex) '5' , 's' , '97' etc.

String Constants:

It is a sequence of characters unquoted in double quotes.

(ex) "Hello" , "1987"

Back slash char constants:

- That are used in output functions.

Constants	***Meaning***
'\a	audible alert (bell)
'\b	back space
'\f'	form feed
'\n'	new line
'\r'	carriage return
'\t'	horizontal tab
'\v'	vertical tab

'\" '	double quote
'\' '	single quote
'\?'	Question mark
'\\'	Back slash
'\0'	null

Variable

It's a data name that may be used to store a data value. It's value can be changed during execution of a program.

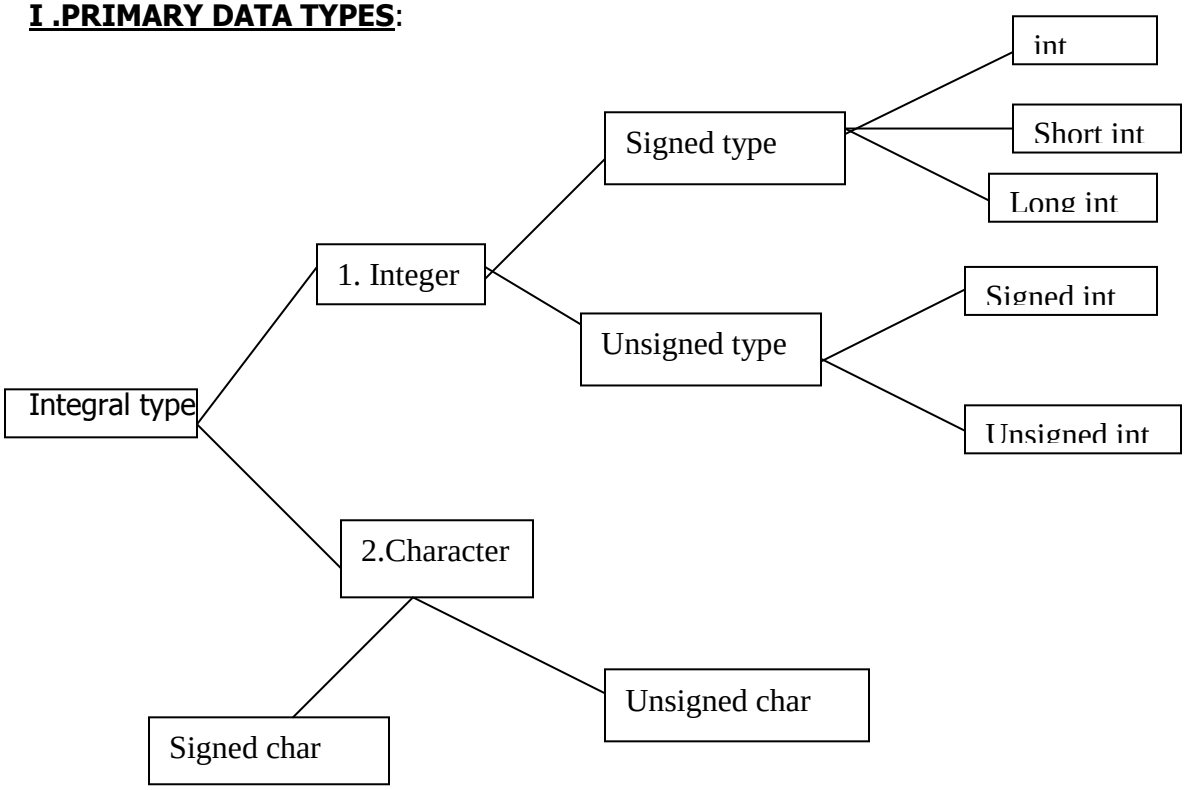
(ex) exam, mark, x1, name 2 etc.,

DATA TYPES

Four classes of data types
Primary data types

1. User Defined Data types
2. Derived Data types
3. Empty Data set

I .PRIMARY DATA TYPES:



Floating Type

i. float ii. Double iii. Long double

II.USER DEFINED DATA TYPE

i. typedef ii. enum

I.typedef:

- That allows user to define an identifier that would represent an existing data type.

Syntax:
typedef type identifier;

- where types refer to an existing data type and identifier refers to the new name given to the data type.
- The existing data type may belong to any class of type.

(ex) typedef int units;
typedef float marks;

units batch;

marks name [50], name 2 [10];

The batch are declared as int variable and name[50], name2[50] are floating point every variables.

- The Advantage of typedef is that we can create meaningful data type names for increasing the readability of the program.

Program:

```
Void main(0
{
typedef int mark;
mark tam,eng,mat;
scanf("%d%d%d",&tam,&eng,&mat);
printf("%d%d%d",tam,eng,mat);
}
```

I. enum

The enum is used to create a new user defined data type.

Syntax:

enum identifier {value1, value 2,……..value n};

The identifier is a user defined data type.

(Ex)
enum day {Monday, Tuesday, …….Sunday};

enum day weak_st, week_end;

week_st = Monday;
week_end = Friday;

Program:

```
Void main()
{
enum mark{tam,eng,mat};
enum mark t,e,m;
t=tam;
e=eng;
m=mat;
scanf("%d%d%d",&t,&e,&m);
printf("%d%d%d",t,e,m);
}
```

Storage classes and their meanings:

i. **auto**

Default is auto, Local variable known to only the function in which it is declare.

ii. **Static**

Local variable that exists and retains its value even after the control is transfer to the calling function.

iii. **Extern**

Global variable known to all functions in the file

iv. **REGISTERED**

Local variable, which is stores in the register is called register variable.

- static & extern variable automatically initialize to zero.

DEFINING SYMBOLIC CONSTANTS

These constants may appear repeatedly in a number of places in the program. It's same times called constant identifiers.

Constant is defines as follows:

#define symbolic-name value_of_constant

(ex)

define STRENGTH 100

define PASS-MARK 50

- same form as a variable names
- No blank space between power sign # and define
- # must not be first char
- must not end with a semicolon
- symbolic name should not be assigned any other value with in the program.
- Not declare Data types

OPERATORS

An operator is a symbol that tells the computer to perform contain mathematical or Logical manipulations. These are many types as given below:

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment/decrement operators
6. Conditional operators
7. Bit wise operators
8. Special operators.

1. Arithmatic operator

- c provides all the basic arithmetic operators

operator	**Meaning**
+	Addition or unary plus
_	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division

(ex)

```
void main( )
{
        int  months, days;
        scanf("%d" , &days);
        months=days /30;
        days=days % 30;
        printf("months = % d \n day = % d" , months, days);
}
```

2. Relational operators

- To compare two quantities and depending on their relation, face certain decisions.

operator	**Meaning**
<	is Less than
<=	is Less than or Equal to
>	is grater than
>=	is grater than or Equal to
==	is equal to
!=	is not equal to

3. Logical operators:

Operator	**Meaning**

&& Logical AND

|| Logical OR

! Logical NOT

4. Assignment operator:

- It's used to assign the result of an expression to variable operator =

(syntax)

V op =expa

- V is a variable
- exp is an expression
- op is a binary Arithmatic operator

(ex)

x + =y +1 same as x = x + (y+1)

x + =3 same as x = x = 3

a = 1

a / = c + b same as a = a / (c+b)

```
#include<stdio.h>
# define N 100
#define A    10
void main()
{
  int a ;

  a = A ;

  while(a<N)
  {
 a * = a;
printf("%d",a);
}
}
```

5.Incrment /Decrement: oparator:

++	Increment operators
_ _	Decrement operators

(ex) m = 5

m ++	same as	m = 6	(post increment)
++ m	same as	m =6	(pre increment)
m - -	same as	m=4	(post Decrement)
- - m	same as	m=4	(pre Decrement)

6. Conditional operators:

- Used to express conditional expression of the form.

Exp1 ? exp2 : exp3;

(ex) a = 10; b = 15;
x = (a>b) ? a: b ;

7. Bitwise operators:

- These are used for testing the bits, or shifting them right or Left
- It may not be applied to float or double

operator	**Meanings**
&	bitwise AND
/	bitwise OR
^	bitwise exclusive OR
<<	shift Left
>>	shift right
~	One's complement

9. Special operators

Operators:
, (Comma operator**)**

Size of operator

Pointer operators **(& and *)**

Member solution operator (**. And →)**

I. Comma operator

It can be used to Link the related expressions together.

(ex) Value = (x=10, y=5, x+y);

ii). sizeof operator

- It's a compile time operator.
- It returns the number of bytes the variable occupies.
- The operant may be a constant, a variable or a data type qualifier.
- It is used to time of determine the length of array and structures when their sizes are not known to the programmer.
- It's used to allocate the memory space dynamically to variables during execution of a program.

(ex)
m = sizeof (sum)
n = sizeof (long int)
u = sizeof (2352)

Program:

```
Void main()
{
char a[10];
char b[sizeof(a)];
scanf("%s",a);
strcpy(b,a);
printf("%s",b);
}
```

Expressions

Combinations of variables, constants, and operators arranged as per the syntax of the Language.

- Integer Arithmetic expressions
- Real Arithmetic expressions
- Mixed mode Arithmetic Expressions

Integer Expressions

The arithmetic operation involving only integer operands is called integer arithmetic expressions.

A – b = 10
A % b = 2

Real expressions:

The arithmetic operation involving only real operands is called real arithmetic expressions.

X = 6.0/7.0 = 0.857143
Z = _2.0/3.0 = _0.66667

Mixed-mode expressions

Combination of real and integer expressions is called mixed mode expressions.

15/10.0 = 1.5

INPUT/OUTPUT operations:

i**) getchar()**

- It is used to reading a single character.

Syntax:
Variable_name = getchar( );

Example:

```
Void main( )
{
char answer;
answer = getchar();
If (answer = 'Y' || answer='y')
printf("OK ");
else
printf("Exit");
}
```

II.putchar( )

- It is used to writing characters one at a time to the terminal.

Syntax:
putchar(variable_name);

Example:

```
void main()
{
char c = 'A';
```

```
putchar(c);
}
```

character Test Function:

Function	Test
isalnum(c)	is c an alphanumeric character?
isalpha(c)	is c an alphabetic character ?
isdigit(c)	is c a digit ?
islower(c)	is c a lowercase
isprint (c)	is c a printable character ?
ispunch(c)	is c a punctuation mare ?
isspace(c)	is c a white space character?
isupper(c)	is c an upper case letter ?

Example:

```
void main ( )
{
char alpha;
putchar('\n');
alpha = getchar();
if (islower(alpha))
putchar(toupper(alpha));
else
putchar(tolower(alpha));
}
```

scanf :

- To input a data line containing mixed mode data.

The general form:

scanf("control strings" , &arg1, &arg2, ……&argn);

- The control string specifies the field format in which the data is to be entered.

(ex) scanf(" % d % c % f ",&a,&b,&c);

Printf:

- To output a data lint containing mixed mode data type

printf ("control string", arg1, arg2, …….argn);

(ex) printf(" % d % c % f",a,b,c);

```
void main()
{
int a;
char b;
float c;
scanf(%d%c%f",&a,&b,&c);
printf(%d%c%f",&a,&b,&c);
}
```

Control strings or I /O Formatted codes:

% c	single character
% d	Decimal Integer
% f	floating point value without exponent
% e	floating point value with exponent
% s	string
% u	unsigned decimal Integer/Address of a variable
% l	signed decimal Integer
% o	octal integer
% X	hexa decimal Integer

prefix for control string :

h	for short integers
l	for long Integers or double
L	for Long double.

DECISION MAKING OR CONTROL STATEMENT:

The decision making statement is used to see whether a particular condition has occurred or not and then execute the certain statements or blocks.

It has three types,

i) Sequential statements
ii) Selection statements
iii) Iteration or Rotation statements

sequential statements :

In this statements are executed sequentially one by one.

(ex)

```
void main ( )
{
```

```
int a, b, c, d, e, i;

scanf("% d% d ", &a, &b) ;

c = a + b;
d = a – b;
e = a * b;
i= a / b;
printf("% d %d % d %d", c, d, e,i);
}
```

II. **SELECTION STATEMENT:**

The order of execution of statement based on certain conditions is called selective statements.

There are on many types,

i. if statement
ii. switch statement
iii. goto statement
iv. conditional operator statement

If STATEMENT

It has different forms,

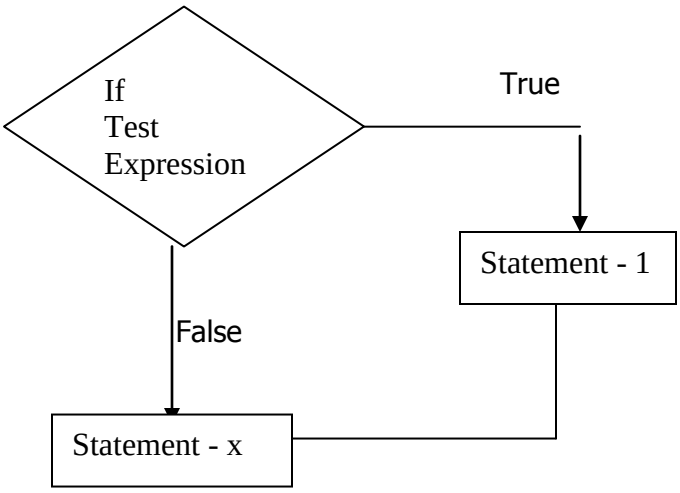
i. simple If
ii. If --- else
iii. Nested if…. else
iv. Else if Ladder

I.Simple if:

The general form :

```
if (test expression )
{
statement  block;
}
statement – x;
```

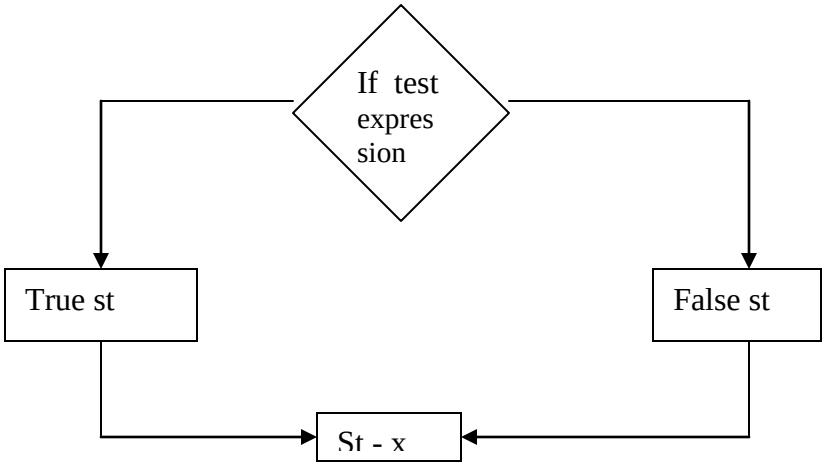
The statement may be a single statement or group of statements. If test expression is true, the statement block will be executed, otherwise, the statement-block will be skipped and the execution will jump to the statement-x.



II. IF -- ELSE statement

```
if (test expression)
{
True-block statement ;
}
else
{
False- block statements;
}
statement  X;
```

If the test expression is true, then the true –block statements are executed., otherwise then the false-block statements executed.



Program:

```
Void main()
{
int a,b;
scanf("%d%d",&a,&b);
if(a>b)
printf(" A IS BIG");
else
printf(" A IS BIG");
}
```

III. Nested IF -- ELSE statement

A multipath decision is a chain of if.

```
If  (condition )
      St – 1;
else  if (condition 2)
      St – 2
else if (condition)
      St – 3
      .
      else
default – st ;
 st - x
```

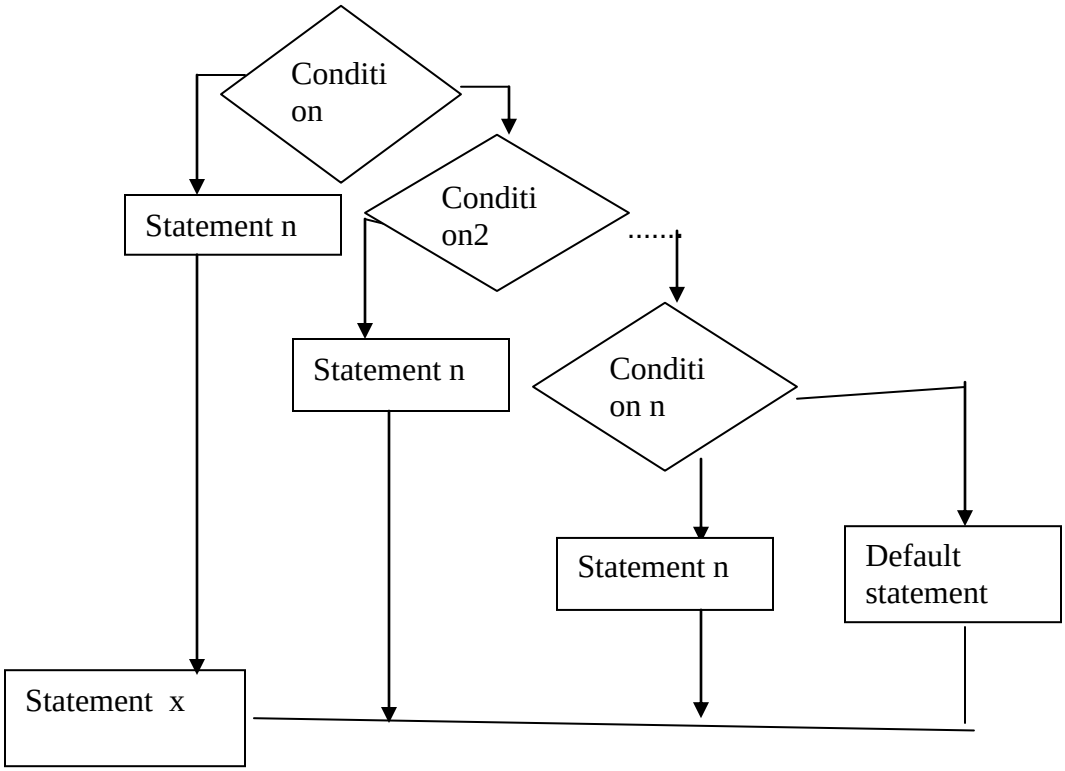
```
if( marks > 79)
Garde = "Honouors";
elseif (marks >59)
Grade = "Firtst Division";
elseif(marks > 49)
   Grade  ="second division";
```

else

Grade = "Fail";

printf(" % s \n" , grade);

IV. ELSE – IF LADDER



(Ex)

colour = "RED"

If (code ! = 2)

Colour = "yellow "

Else
Colour = "white"

Else
Colour="green"

Program:

Void main()
{

```
int month,day,year, feb;

printf("ENTER THE MONTH");
scanf("%d",&month);
if(month==2)
{
        printf("MONTH IS FEBRUARY");

        if((year % 4) == 0)
        printf("LEAP YEAR");
        feb=29;
        }
        else
        {
        printf("NOT A LEAP YEAR");
        feb=28;
}
printf(MAXIMUM DAYS IN THE MONTH: %d", feb);
}
}
```

Iteration Statement:

i.**While Statement:**

- The while is an entry-controlled loop statement.
- The test condition is evaluated and if the condition is true, then the body of the loop is executed.
- In this process of repeated execution of the body continues until the test condition is becomes false.

The General Form:

```
While (test-condition)
{
body of the loop;
}
```

Program:

```
Void main()
{
int I=0;
while(I<=10)
{
printf("%d",I);
I++;
}
}
```

ii. Do-While statement:

- It's an exit-controlled loop statement.
- In this do-while, to evaluate the body of the loop is first. So, the body of the loop is always executed atleast once.
- If the condition is true, the body of the loop is executed. Otherwise the loop is terminated.

The General Form:

```
do
{
body of the loop;
}
While (test-condition)
```

Program:

```
Void main()
{
int I=0, sum=0;
do
{
s=s+I;
printf("SUM OF SERIES=%d",I);
I++;
} while(I<=10)
}
```

for loop:

- A for loop is an entry-controlled loop statement.

The General Form:

```
for (initialization; test-condition; increment/decrement)
{
body of the loop;
}
```

The execution of for statement is as follows:

i. An initialization of the control variable is done first.

ii. The value of the control variable is tested using the test condition.
   - If the condition is true, the body of the loop is executed. Otherwise the loop is terminated.

iii. To increment or decrement the control variable until the value of control variable is fails to the test condition.

Program:

```
Void main()
{
```

```
Int I, sum=0;
for (I=1;I<=20;I+=2)
{
sum = sum + I;
}
printf(" SUM OF ODD NUMBERS=%d",sum);
}
```

Break:
Syntax:
break;

- When a break is encountered inside a loop, the loop is immediately exited.
- It's will exit only a single loop.

Program:

```
void main()
{
int k;
for(k=1;k<10;k++)
{
if(k= =5)
break;
printf("k value=%d\n",k);
}
}
```

In this program, the k value is displayed up to 4 and the loop is terminated.

Continue statement:

A continue statement causes the particular loop to be continued with the next iteration after skipping any statements in between the continue statement skip the following statements and continue with the next iteration.

Syntax:
continue;

Program:

```
void main()
{
int k;
for(k=1;k<10;k++)
{
if(k= =5)
continue;
```

```
printf("k value=%d\n",k);
}
}
```

In this program, the k value is displayed up to 9, except the value 5 is not displayed.

SWITCH STATEMENT

- It ' s built in multiway decision statement.
- To test the value of a given variable for the list of values and when match is formed, the block of associated case statements are executed.

```
switch (expression)
{
case value- 1 :
block-1 ;
break;

case value – 2:
block–2
 break;
……..
……..
……..

default:
default block;

break;

}

statement – x;
```

The break is used to exit the from the switch statement.

The default is optional. If the value of expression doesn't match, the default block is executed.

II. Conditional statement:

_ It is useful for managing to way decisions.

General form:

Conditional_expression ? exp1 : exp2

If the conditional expression is non zero, exp1 is evaluated otherwise exp2 is evaluated.

(Ex)

```
void main()
{
int big,a,b;
scanf("%d%d",&a,&b);
big = (a<b), ? a: b;
printf("%d".big);
}
```

I. goto st

_ To branch unconditionally from one point to another in the program.

Statement ;

- A goto break the normal sequence execution of the program.
- if the label is places after goto label.
- Some statements will be skipped and the jump is known as forward jump.
- If the label is before the go to statement; a loop will be formed and some statements will be executed repeatedly. It's known as backward jump.

(ex)

```
void main ( )
{
double x, y;

read:
scanf(" % f", & x );

if ( x<0) goto  read;
y =sqrt(x);
printf (" % f % f " , x, y);

goto  read;
}
```

ARRAYS

An Array is a group of related data items that share a common name.

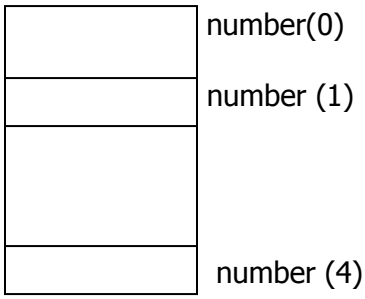
The complete set of values is referred to as an array; the individual values are called elements.

There are three types of arrays.

i. One Dimensional Array or Single Dimensional Array.

A list of items can give one variable name using only one subscript and such a variable is called a single dimensional array.

(i.e) int number[5];



Declaration:

Data_type variable [size];
(ex) float height [50];

```
void main ( )
{
int a[5];

for (i = 1; i<=5; i++)                                    {

scanf (" % d", &a[i]);

for(i =1;i<=5; i++)

printf (" % d", a[i]);

}
```

Two Dimensional Array:

C allows as defining such tables of items us being two dimensional arrays.

Declaration

Type Array_name [row size] [column size];

(ex) int v[2] [2];

a[0,0], a[0,1], a[0,2]
a[1,0], a[1,1], a[1,2]
a[2,1], a[2,1] ,a[2,2]

	column 0	column 1	column 2
Row 0	[0] [0]	[0] [1]	[0] [2]
Row 1	[1] [0]	[1] [1]	[1] [2]
Row 2	[2] [1]	[2] [2]	[2] [2]

MATRIX ADDITION

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad b = \begin{pmatrix} 9 & 8 & 7 \\ 6 & 8 & 4 \\ 3 & 2 & 1 \end{pmatrix}$$

```
void main ( )
 {

int i, j, k, a[2][2], b[2] [2], c[2] [2];

for (i =0; i<2; i++)
for (j = 0; j<2; j++)
{
scanf (" % d" , & a[i] [j]);
}

for (i =0; i<2; i++)
for (j =0; j<2; j++)
{
scanf (" % d" , &b [i][j]);
}
```

```
for(i =0; i<2; i++)
for(j=0; j<2; j++)
{
c[i] [j] =0;
c[i][j] = a[i][j] + b[i] [j] ;
}

for (i =0; i<2; i++)
for (j =0; j<2; j++)
{
printf(" % d" ,c[i][j]);
}
}
```

INITIALIZING TWO DIMENSIONAL :

Static int table [2] [3] = {0,0,0,1,1,1};

MULTI DIMENSIONAL ARRAYS:

C allows arrays of three or more dimensional. The general term of a multidimensional Array is,

Type arrays name [s1] [s2] [s3]……[sn];

String:

A string is an array of characters.
Any group of characters defined between double quotation marks is a constant string.
Ex:
"WELL DONE"
"1234"

Declaring and Initializing strings:

A string variable is any valid c variable name and is always declared as an array.

The General Form:

char string_name[size];

- The size determines the number of characters in the string name.
- When the compiler assigns a character string to a character array, it automatically supplies a null

character(\0) at the end of the string.

- The size should be equal to the maximum number of characters in the string and plus one.

Example:

char city[10];

- The character may be initialized when they are declared.

static char city[5]=”CHENNAI”;

OR
static char city[5]={‘C’, ’H’, ‘E’, ‘N’, ‘N’, ‘A’, ‘I’};

Reading Strings:

- The input function scanf can be used with %s to read in a string of characters.
- The writing function printf can be used with %s to writing a string to terminal.
- The problem with the scanf functions is that it terminates its input on the white spaces.

```
Void main()
{
char name[15];
scanf(‘%s”,name);
printf(‘%s”,name);
}
```

Reading a Line of Text:

- It is not possible to use scanf function to read a line containing more than one word.
- An entire line of text can be read and stored in an array used by getchar() function.
- This functioning is a reading a character repeatedly until the new line(‘\n’) character is encounted.

```
Void main()
{
char line[50];
int c;
c=0;
while(char!=’\n’)
{
char = getchar();
line[c]=char;
c++;
}
line[c]=’\0’;
printf(“%s”,line);
```

}

Function	Action
strcat( )	Concatenates two strings
strcmp( )	Compares two strings
strcpy( )	Copies one string over another
strlen( )	Finds the length of a string

strcat

The strcat functions joints two strings together.

The general form:

strcat(string1, string2);

Example:

```
void main()
{
char a[10],char b[10];
scanf("%s",a);
scanf("%s",b);
strcat(a,b);
printf("%s",a);
}
```

strcmp

If two strings are equal and has a value equal to zero, otherwise it has difference numeric values.

Syntax:
strcmp (string1,string2);

Example:

```
#include<stdio.h>
void main()
{
char a[10],char b[10];
scanf("%s",a);
scanf("%s",b);
n=strcmp(a,b);
if(n==0)
```

```
printf("strings are equal");
else
printf("strings are not equal");
}
```

strcpy()

It is used to copies one string over another.

Syntax:

```
strcpy(string1,string2);
```

```
#include<stdio.h>
void main()
{
char a[10],char b[10];
scanf("%s",a);
scanf("%s",a);
scanf("%s",b);
strcpy(a,b);
printf("%s",a);
}
```

strlen()

If returns the number of character is a string.

Syntax:

```
n = strlen(string);
```

- n is a integer variable

Example:

```
#include<stdio.h>
void main()
{
char a[10],int b;
scanf("%s",a);
b=strlen(a);
printf("%d",b);
}
```