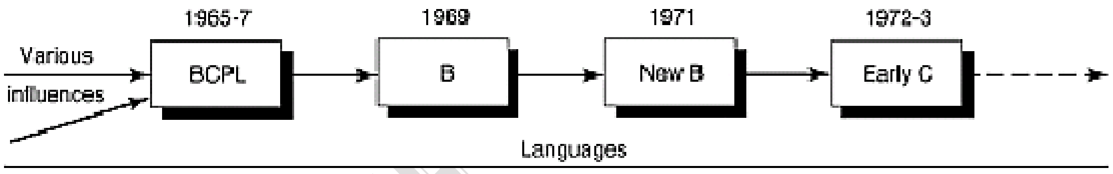
History of 'c'

'C' is a powerful programming language, which is attracting attention world wide because the software industries are adopting the language to great advantage one reason for this popularity. That is to say, a program written in 'c' can be transferred easily from one computer to another computer with minimal changes or not at all. Programs written in 'c' are fast and efficient. This versatility makes 'c' a desirable language in the highly competitive software industries.

A system programmer named Dennis Ritchie developed the 'c' language at bell labs in early 1972. It was written originally for programming under an operating system called Unix. This language is derives from the fact. That is based on an earlying version written by ken Thomson, another bell labs system engineer. He adopted it from a language that was known by the initials **B.C.P.L** that stands for Basic Combined Programming Language. To distinguish his version of the language from **B.C.P.L,** Thompson dubbed 'b'. The first of the initials **B.C.P.L** when the language was modified and improved to its present state. The second letter of **B.C.P.L** 'c' was chosen to represent the new version.



Character set of 'c'

'c' uses the following set of characters.

Letters: A – Z, a – z.

Digits: 0 – 9.

Special characters: *, (,), #, &, [,], ', ", +, -, <, >, … etc.

Identifiers and key words

Identifiers are names given to various program elements. Such as variables, functions and arrays. Identifier consists of letters and digits and in any order except that the first character must be a character, both upper and lower case letters are allowed. The under score character can also be used in the identifiers. In most cases, this character is used in the middle of the identifier. An identifier may also begin with under score character.

Examples:

Valid	**not valid**
Temp	8level
_78po	cust name

Key words (or) reserve words:

The reserve words of ‘c’ are called keywords. These key words will have predefined meanings and these can’t be used as user defined identifiers. The standard key words are

auto	**break**	**case**	**char**	**const**
continue	**continue**	**default**	**do**	**double**
else	**enum**	**extern**	**float**	**for**
goto	**if**	**int**	**long**	**register**
return	**short**	**signed**	**sizeof**	**static**
struct	**switch**	**typedef**	**union**	**unsigned**
while				

Note: the all reserve words must be in lower case letters.

Data types

The basic data types of ‘c’ are

1. **int :** This type of data represents an integer quantity and the memory requirement for an integer is two bytes.
2. **char :** This type of data represents a single alphabet characters and it requires on byte of memory.
3. **float :** This type of data represents a float point number. That is a number containing a decimal point and it can also be represents in exponent form. The memory requirement is four bytes.
4. **double :** This data type represents double precision floating point number. The memory requirement is 8 bytes.

Some data types may also contain data types qualifiers short, long, unsigned. These are used for memory adjustment and they can be represented as short int, long int, unsigned long int, unsigned short int.

Data type	**size (bytes)**	**range**
Char or signed char	1	-128 to 127
Unsigned char	1	0 to 255
Int or signed int	2	-32768to 32767
Unsigned int	2	0 to 65635
Short int or signed short int	1	-128 to 127
Unsigned short int	1	0 to 255
Long int	4	2,147,483,648to 2,147,483,647
Unsigned long int	4	0 to 4,294,967,295
Float	4	3.4e-38 to 3.4e+38
Double	8	1.7e-308 to 1.7e+308
Long double	10	3.4e-4932to 1.1e+4932

Constants

'c' has four basic types of constants.

1.integer constant: the integer constant represents number and are called as numeric type constants. The following rules to applied all numeric type constants.

A. The constant can be preceded by - sign if desired.

B. Camas and spaces cannot be included with in the constant.

C. The value of constant cannot be exceed minimum and maximum bends specified by the compiler.

2. floating point constant: a floating-point constant is a base ten number that consists ei9ther a decimal point or an exponent or both. The exponent must be an integer.

Example: 0.1, 0.01, 20.0, 0.2e-1, 0.2e+2.

3. character constant : a character constant is single character enclosed in a single quotation or obstructs**.**

4. string constant : a string constant is a group of letters , we can use either upper or lower case letters.

Example: 'a', 'b', 'c' constant is consists of any number of characters enclosed in double quotation marks.
"srinivasarao".

Escape sequences

Certain non-printing characters as well as the double quotation the quotas. The question mark and the back slash can be expressed inters of escape sequence. An escape sequence is always begin with a backward slash (\ ) and is followed by one or more special characters.

Character	escape sequence	ASCII Value
Bell	\a	007
Backspace	\b	008
Horizontal tab	\t	009
Vertical tab	\v	011
New line	\n	010
Form feed	\f	012
Carriage return(enter key)	\r	013
Question mark	\?	063
Quotation mark	\"	034
Single quote	\'	039
Back slash	\\	092
Null	\0	000

Variables & arrays

A variable is an identifier that is used to represent some specified type information. Variable represents a single data item i.e. a numeric quantity or character constant.

Example: sum, product, total, average.

The array is another kind of variable used in 'C' an array is an identifier that refers to a collection of date items which all have the same name. All data items must be of the same type. They array elements are refereed by using subscripts. Each array variable must be followed by a pair of square brackets ([ ]) Containing a positive integer, which specifies the size of the array.

If the array contains ' n ' elements then the subscript must be an integer quantity whole value ranging from 0 to n-1. An ' n ' character string requires n+1 element array because if the null character automatically placed at the end of string.

Example: a[5] H E L L O O

a[0] to a[4]

Character type array is used to represents a string.

Each array element represents one character with in the string.

Declarations:

The entire variable must be declared before. They are using in executable statements. A declaration consists of a data type followed by one or more variable names ending with a semicolon [;]. A pair of square brackets containing a positive integer. Which specifies the size of the array must follow which array variable. The syntax for these declarations is

Data type variable listing

(or)

Data type variable1, variable2, variable3,-------------etc;

```
Example:  int a, b, c;
          Char sum;
          Int a [5];
```

Initial values can be assign to variables with type declaration. If declared, the declaration must consist of date type followed by a variable name, an equal sign and a constant of an appropriate type.

```
Example: int value=5;
         Char name [10] = "srinivas";
```

Operators and expressions

Operators: There are different types of operators in ‘ C’ languages. Those are

A. Arithmetic operators: There are five arithmetic operators they and in ‘C’ is

Operator	**meaning**
+	addition
-	subtraction
*	multiplication
/	division
%	modulus

(remainder after integer division)

EXAMPLE : int a=10, b=5;

a + b = 15, a – b = 5,

B.**Relational operator:**

Operator	**meaning**
<	less than
>	greater than
<=	lessthan or eual
>=	greater than or equal
= =	equal to
! =	not equal

C. Logical operators:

Operator	**meaning**
&&	and
I I	or
!	not

D. Assignment operators: the assignment operator symbol is = and this operator syntax is

Identifier = expression

Example: a =10, a[5] =20, c = a + b

The following are some more assignment operators in ‘ C ‘

+ = , - = , % =, / =

Expression 1 + = expression 2;

(Or)

Expression 1 = expression 1 + expression 2 ;

Examples: a + =2 or a= a +2

a - =2 or a= a –2

a * =2 or a= a *2

E. Unary operators: operators, which act up on a single operand to produce a new value, are called unary operators. The unary operators are

+ + , - - , size of

EXAMPLE: I + + or I = I + 1

I - - or I = I – 1

Size of [I] 2 bytes

Statements

There are three types of statements in ‘ C ‘ languages. The statements cause the computer to carry out some action. The three type statements are

1. Expression statement 2. Compound statement 3. Control statement

1. **Expression statement:** an expression statement consists of an expression followed by a [;] semicolon. The expression is evaluated when the statement is executed.
   Example: c = a + b;

2. Compound statement: this statement consists of several statements enclosed with in a pair of flower brackets i.e. {} the several statements may be expression statement, compound statement, and control statement.

Example:

```
{
  Temp = a ;
  a = b ;
b = temp;
        }
```

3. **Control statement:** these are used to create special program future such as logical test loops and branches.

Example: while, do whille, for, if , else.

INPUT AND OUTPUT STATEMENTS

Single character- input – getchar ( ) :

A single character can be entering from the input device to the computer using getchar (). It is a part of I/O library. This function does not require any arguments. The general form of the getchar ( ) is.

Syntax: data type variable = getchar ();

Example: char c= getchar ( );

Where character variable is a variable declared a character type.

Single character – output – putchar ( ) :

Single character can be displayed using the library function putchar (). It transmits single character to standard output device.

Syntax: putchar (character variable);

Example: putchar (a);

CONVERSION CHARACTERS

% c	character
% d	integer
% f	float
% e	exponent
% s	string
% x	hexadecimal
% o	octal

The conversion character indicates the type of corresponding data item. The separate group of items is separated by blank space.

The arguments may be variables, arrays whose type must match the corresponding character groups in the control string. Each variable name must be proceeded by an '&' except array names. It indicates that the arguments in an address indicating the data in the memory where the argument is placed i.e. address of location.

1.write a program to read a single characters and displays the output.

```
# include < stdio.h >
main ( )
{
char a ;
clrscr ( );
a = getchar ( );
putchar ( a );
getch ( );
}
```

SYMBOLIC CONSTANTS

A symbolic constant is name that substitutes for a sequence of characters. The characters may represent numeric constants, character constants or string constants. Thus a symbolic constants allows a name to appear in place of numeric constants or a character constant or a string constant. Symbolic constants are defined at the beginning of the programming and the general form is

Syntax: # define <NAME> text

EXAMPLE: # define MAX 20

Where ‘ NAME ‘ indicates the symbolic name written in upper case letter and text represents the sequence of characters associated with a symbolic name and the text does not end with (;). Because it is not a true statement of ‘ C ‘ . Moreover it text to end with a (;) this (;) would be created although it were a part of symbolic constant.

The substitution of the text for a symbolic constant is done anywhere in the program except with in a string.

Writing output data:

Printf ( ) : output can be written from the computers on to an output device using library function printf ( ). This function can be used to output any combination of numerical values single character and strings.

Syntax: printf (control string, arg 1, arg2, arg3…);

Where the control string is a string that contains formatted information and arg1, arg2…etc. arguments that represents the separate data items the arguments can be functions, arrays, single data items.

Writing input data:

Scanf ( ):the input can be entered into the computer using scanf ( ). The functi0on can be used to enter any combination of numeric values, single character and strings. The syntax of scanf function is

Syntax: scanf (control string, arg1, arg2, arg3…);

The control string differs to a string containing some formatted in formation and arg1, arg2 etc represents the separate input data items. The control string contains separate group of items for one character for each input.

2. Write a program to read 3 integer values and displaying three values.

```
#include < stdio.h >
main ( )
{
int a , b , c ;
clrscr ( );
printf ( “ enter a , b , c values “ );
scanf ( “ %d %d %d “ , &a,&b,&c );
printf ( “ a=%d b=%d c=%d “, a , b , c) ;
getch ( );
}
```

3. Write a program to read the three values and calculate the sum and average display the sum and average.

```
# include <  stdio . h>
main ( )
{
float a, b, c, sum = 0, avg = 0;
clrscr ( );
printf ( ‘’  enter the three values ‘’  );
scanf  ( ‘’ %f %f %f ‘’ , &a , &b , &c );
sum = a + b + c;
avg = sum / 3 ;
printf ( “ sum = %f “ , sum );
printf  ( “ avg = %f “ , avg ) ;
getch ( ) ;
}
```

4. write a program calculate the ax2+bx+c.

```
# include <  stdio . h>
main ( )
{
float a, b, c, x, sum =0;
printf ( “ enter a, b, c, x “ );
scanf ( “ %f %f %f %f “, &a, &b, &c, &x );
sum = ( a*(x*x)) + (b*x) + c;
printf ( “ the sum is %f “, sum );
getch ( );
}
```

unsolved problems

1. **Write a program to calculate the simple interest and display the result.**
2. **Write a program to calculate the area (3.1416*(r*r)), circumference of circle (2*3,1416* r) read radius and display result.**
3. **Write a program to calculate the net salary. (basic=1600,D.A=7%, H.A=2%,T.A= 10%).**

IF ELSE STATEMENTS

The if statement is another control statement which is used for branching. This branching is depends on a condition given along with if statement. This statement can be used to define to different groups of statements out of those two groups the one group of statements can be execute in each time the section of group of statements. If the condition is true the first group of elements i.e. the statements between if and else gets executed. If condition is false the second group of statements will be executed. The syntax of the statement is

Syntax: if < condition >

```
{
Statements
…
…
}
Else
{
Statements
…
…
}
```

5. Write program to read one number and check the number is even or odd

```
# include < stdio. h>
main ( )
{
int n;
clrscr ( );
printf ( " enter a number " );
scanf ( " %d " , &n);
if ( n%2 == 0 )
{
printf ( "the number is even number " );
}
else
{
printf ( " the number is odd number " ) ;
}
getch ( );
}
```

6. Write a program to check the given year is leap or not and display the result.

```
# include < stdio.h >
main ( )
{
```

```
int y;
clrscr ( );
printf ( “ enter the year “ );
scanf ( “%d “ , &y);
if ( y %4 = = 0)
{
printf ( “ the year is a leap %d “,  y);
}
else
{
printf ( “ the year is not leap year %d “, y);
}
getch( );
}
```

WHILE STATEMENT

The while statement is used to carry out looping operations. The syntax of the statement is

```
While < condition >
{
Statement 1;
Statement 2;
}
```

The statement can be executed repeatedly as long as the value of the condition expression is true. The statement 1, statement 2, etc, can be a simple or compound statement.

7. Write a program to accept minimum and maximum values and calculate the sum between minimum and max values.

```
# include < stdio.h >
main ( )
{
int mi, ma, sum =0;
clrscr ( );
printf ( “ enter the minimum value and maximum value “ );
scanf ( “%d %d “, &mi,&ma);
while ( mi<= ma)
{
sum= sum+mi;
mi++;
}
printf( " the sum is %d",sum);
getch ( );
}
```

8. **Write a program to calculate the area and circumference. If the radius less than zero the program will be terminated.**

```
# include < stdio. h >
main ( )
float pi=3.1416,r,a=0,c=0;
clrscr ( );
printf ( " enter the radius " );
scanf ( "%f ",&r);
while ( r > 0 )
{
a=pi*r*r ;
c=2*pi*r ;
printf ( " area %f circumference %f " ,a,c);
getch ( );
clrscr( );
printf ( " enter the radius " );
scanf ( " %f " , &r );
}
getch ( );
}
```

9. **Write a program to enter any value and check it is prime or not.**
[Example 2, 3, 5, 7,11,13…]

```
# include < stdio.h>
main ( )
{
int c=0, n,i=1;
clrscr ( );
printf ( " enter any value ");
scanf ( "%d ",&n);

while (i < = n)
{
if (n%i ==0)
c+ +;
i + +;
}
if ( c = = 2)
printf ( " this is prime number " );
else
printf ( " this is not prime number " );
getch ( );
 }
```

Unsolved programs

1. **write a program to calculate the between 1 to 50 even numbers and display the sum,**
2. **Write a program to read the individual digit and display the sum of the individual digit.**
3. **write a program to read the individual digit and display the reverse order**
4. **write a program to accept a number and display the multiplication table.**
5. **Write a program to display the multiple tables 4 to 8.**
6. **Write a program to calculate the factorial values between 2 to 7.**
7. **Write a pro9gram to calculate the perimeter and area until one of the length and width is negative the program will terminate [p= 2 (l+w),a=l*w].**
8. **Write a program to calculate the sum of the odd and even numbers between lowest and highest values.**
9. **Write a program to print all numbers, which are dividing by four between1 to 50.**

DO WHILE

The do while statement is also similar to the while statement can also used for looping purpose that means by using this statement also can execute a set of statements repeatedly. The syntax for this statement is

Syntax: do

```
        {
Statements;
…
}
While  < condition > ;
```

The statement will be executed repeatedly as long as the value of condition i9s true. The statements are executed at least once. The statements can be either simple or compound statements.

10. Write a program to calculate the area and circumference. If the radius less than zero the program will be terminated. Using the do while.

```
# include < stdio.h >
# define pi 3.1416
main ( )
{
```

```
float r, c = 0, a = 0;
clrscr ( );
do
{
printf ( “ enter the radius “ ) ;
scanf ( “ %f “, &r );
c = 2 * pi * r ;
a = pi * r * r ;
printf ( “ circumference = %f area = %f “, c,a );
}
while ( r > 0 );
printf ( “ the value is not correct “ );
getch ( );
}
```

11. Write a program to find out sum of the positive integers given value the program has to terminate the number.

```
# include < stdio.h >
main ( )
{
int n,s=0 ;
clrscr ( );
do
{
printf ( “ enter the value “ );
scanf ( “ %d “, &n );
s = s + n;
}
while ( n > 0 );
printf ( “ the sum is %d “ , s );
getch ( );
}
```

FOR STATEMENT

The for statement is most commonly used looping statement.

This statement includes an expression that spacious an initial value for an index, another expression determines whether or not the loop is continued and a third expression that allows the index to be modified at the end of the each statement.

The statement can be a simple or compound statement. The syntax of the for statement is

For (expression 1; expression 2; expression 3)

Statements;

Where expression 1 is used to initialize some parameter that controls the looping action expression 2 represent a condition that must be satisfied for the loop to continue execution and expression 3 is used to after the value of the parameter initialized by expression. So, expression 1 is an assignment expression, expression2 is a logical expression, expression3 can be unary expression or assignment expression.

When a for statement is executed expression2 is evaluated and test before each time the loop is executed and expression3 is evaluated at the end of the loop the looping action will continue a long as the value of expression2 is true.

12. Write a program to print sum of integers between low value and high value.

```
# include < stdio.h >
main ( )
{
int i, h, sum = 0;
clrscr ( );
printf ( “ enter the low and high value “ );
scanf ( “ %d %d “ ,&l, &h);
for ( i= 1;i<= h;i++ )
{
sum = sum + i ;
}
printf ( “ the sum of low between high values %d”,sum );
getch ( );
}
```

13. Write a program to generate the numbers decreasing order.

```
# include < stdio.h >
main ( )
{
int n, i;
clrscr ( );
printf ( “ enter the number “ )’
scanf ( ‘ %d “, &n );
for ( i=n; i> 0; i - -)
{
printf (“ %d “, i );
}
getch ( );
}
```

14. **Write a program to generate the following series**

1

1 2

1 2 3

1 2 3 4.

```
# include < stdio. h>
main ( )
{
int n, i, j ;
printf ( “ enter the number “ );
scanf ( “ %d” , &n );
for ( i =1;i< n; i ++)
{
for ( j=1; j< n ;j ++)
{
printf ( “%d”, j );
}
printf ( “ \ n “ );
}
getch ( );
}
```

15.write a program to generate the following series

1

1 2

1 2 3

1 2 3 4

1 2 3

1 2

1

```
# include < stdio.h >
main ( )
int i, n, j ;
printf ( “ enter the n value “ );
scanf ( “ %d” ,&n);
for ( i = 1; i < = n ; i + +)
{
for ( j = 1; j < = i ; j + +)
{
printf (“ %d ”,j );
}
printf ( " \ n " );
}
for ( i = (n- 1); i>=1; i- -)
{
for ( j =1;j<= i;j++)
{
printf ( "%d",j);
}
printf ( " \ n ");
}
getch ( );
}
```

16.write a program to generate the following series

```
    1
  1 2 3
1 2 3 4 5
```

```
# include < stdio.h >
main ( )
int i,n, j, r, c;
clrscr ( );
r = 12;
c = 40;
printf ( " enter the n value ' );
scanf ( " %d ', &n);
for ( i = 1; i<=n; i+=2)
{
gotoxy ( c, r);
for (j =1;j<= i; j ++)
{
printf ("%d",j);
}
r=r+1;
c= c – 1;
printf ( " \ n" );
}
getch ( );
}
```

17.write a program to generate all the prime numbers between 1 to 100

```
# include <stdio.h >
main ( )
{
int i = 1,c,j,r ;
while ( i<= 100)
{
c=0;
for (i =1;j<= i;j++)
{
r = i % j ;
if ( r = = 0)
c + +;
}
if (c == 2)
{
printf ( " %d " , i );
}
i + + ;
}
getch ( );
}
```

<u>unsolved programs</u>

1. **Write a program to repeat the calculations 6 times.**
2. **Write a program to repeat the calculations specified number of times using for loop.**
3. **Write a program to calculate the some of integers low through high with an exchange is necessary.**

Array

In ' C ' as in other computer language it is assign to possible single name to a whole group of similar data for example, if we are interested in a large number of recorded Cecil's temperatures. We can assign common names such as a temp to all of the data. Then we can refer to each element in terms of its position with in the list of items. This done in mathematics as well as in computer programming, using an entity called an array.

An array is a group of elements that sto0re a common name and that are differentiated from one another by their positions with in the array. For example if we have five numbers, all of which are named a, they may be listed.

A
10
20
30
40
The position of each of these elements can be indicated by means of a subscript.
A0=10
A1=20
A2=30
A3=40
In mathematics a subscript is a number written to the right of the variable name, slightly below the line, and usually in small type. Since, it is impossible to display subscript numbers on the standard computer terminal. The usual records are to enclose the subscript in bracket. In ‘c’ square brackets are used. The following five statements are valid in ‘c’.
a[0]=10   a[1]=20   a[2]= 30   a[3]= 40

Declaring an array

An array must be declaring, since it is a type of variable. An array contain five elements, all of five which are integers can be declared as follows
int x{5];
An array name must be choosing according to the same rules used for naming any other variables. The size of the array is specified usage the subscript notation, in our example, the subscript 5 indicates how many elements are to be allowed to array x.
The method of subscripting arrays in ‘c’ is different from the used in much other programming language. The elements of a five-element array in ‘c’ are numbered starting with one. Therefore, the assignment statement for this example actually should be written as follows.
X[0]=10 x[1]=20   x[2]=30   x[3]=40

Initializing an array

The array can be initialize as a global as you will recall, a global variable is one that is declared out side any function (usually the main( )). It consists through out the execution of the p0rogram, and is useful from any function with in the program. A special future of global arrays is that they can be initialized when they are declared this done by following the array name and dimension with an equal sign, followed by a pair of braces ({ })these braces contain a series of constant values separated by commas.
Example: int a[3]={5,10,15};

18.write a program to input any five integers into an array and print them.

```
# include < stdio.h>
main( )
{
int a[5],i ;
clrscr( );
for ( i =0; i<5;i++)
{
printf ( ‘enter the digit #%d”, i+1);
scanf( “%d”, &a[i] );
}
for (i=0;i<5;i++)
{
printf (“ #%d %d\n”, i+1.a[i]);
}
getch ( );
}
```

19.Write a program to input any five integers and print them in reverse order.

```
# include <stdio.h>
main ( )
{
int a[5],I;
clrscr ( );
for (i=0;i< 5;i + +)
{
printf (“ enter the digit #%d”, i+1);
scanf (“%d”,&a[i]);
}
printf ( “ the reverse order is”);
for (i=5-1;i>=0;i--)
{
printf (‘ %d”,a[i]);
}
getch( );
}
```

20.write a program to sort the values of an array by entering specified numbers into an array.

```
# include < stdio.h>
main( )
{
int a[30],I,n,j;
clrscr ( );
printf(" enter how many numbers");
scanf ( "%d",&n);
for (i=0;i< n;i++)
{
printf ("enter the %d number",i+1);
scanf ("%d ",&a[i]);
}
for (i=0;i< n;i+ +)
{
for(j=i+1;j<n;j+ +)
{
if ( a[ i ] >a [ j ] )
{
int temp;
temp = a[i];
a[ i ]=a[ j ];
a [j] = temp;
}
}
}
printf (" the sorted order is ");
for (i = 0; i < n;i + +)
{
printf ("%d", a[i]);
}
getch( );
}
```

Unsolved programs

1. **Write a program to initialize the array as a global with in the given data 5, 15,25, 35, 45. Print that array in original order and reverse order.**
2. **Write a program to read the values in to an array by defining the size with symbolic constant and print those values on the screen.**

3. **Write a program to store even numbers into one array odd numbers into another array using the numbers 1 to 20.**
4. **Create a 3 array variables containing 5 elements in each enter desired values to first two arrays and calculate the addition of the corresponding element and store those in the third array. Print three arrays.**
5. **Write a program to accept 10 numbers and calculate average. Which value is an above the average and print that value.**
6. **Write a program to accept string information in to an array and print that string in reverse order.**

MULTI DIMENSIONAL ARRAYS

The elements of two-dimensional arrays are located by means of a row and column, so row subscripts are required a two dimensional array. The row subscripts are specified before the column subscript in two dimensional array x[i] [i]

X[i] represents a row and x[i][i] represents a column with in that row 'i'. The value can be read in to two-dimensional array.

Syntax: data type name [rows][columns];

Example: int a[10][10];

21.**Write a program to read one matrix size is 2*3 and displays the matrix.**

```
# include<stdio.h>
main()
{
int i,j,a[20][20];
clrscr( );
printf("enter the values");
for(i=0;j<2,i++)
{
for ( j=0;j>3;j++)
{
scanf("%d",&a[I][I]));
}
}
printf {"the a matrix is \n");
for(i=0;i<2;i++)
{
for (j=0;j<3;j++)
}
printf("%d", a[i][j]);
}
```

```
printf("\n");
}
getch();
}
```

22.write a program to read two 2 * 2 matrix and calculate sum of the two matrixes and display the sum of the matrix and matrixes.

```
# include<stdio.h>
main()
{
int  i,j,a[20][20],b[20[20], c[20][20];
clrscr( );
printf("enter the values in A matrix");
for(i=o;i< 2;i++)
{
for (j=0;j<2;j+=)
{
scanf("%d",&a[ i ][ j ] );
}
}
printf ( " enter the values in B matrix ");
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 ]=a[ i ][ j ]+b[ i ][ j ];
}
}
printf("sum of the two matrixs\n");
for(i=0;j< 2;i++)
{
```

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

23. write a program to accept the two 2*2 matrixes and calculate the multiplication of matrix and display the three matrixes.

```
# include<stdio.h>
main( )
{
int a[20][20],b[20][20],c[20][20],i,j,k ;
clrscr( );
printf("enter the A matrix values" );
for( i=0;i<2;i++)
{
for( j=0;j<2;j++)
{
scanf("%d",&a[ i ][ j ] );
}
}
printf("enter the B matrix values" );
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;
for(k=0;k< 2;k++)
{
c[ i ][ j ]=c[ i ][ j ] + a[ i ][ k ]*b[ k ][ j ];
}
}
}
printg("the A matrix is \n");
for( i=0;i< 2;i++)
{
j=0;j<2;j++)
{
printf("%d", a[ i ][ j ] );
}
printf("\n");
}
printf("the B matrix is\n");
for( i=0;i<2;i++)
{
j=0;j<2;j++)
{
printf("%d",b[ i ][ j ] );
}
printf("\n");
}
printf("the C matrix is\n");
for(i=0;i< 2;i++)
{
for( j=0;j< 2;j++)
{
printf(" %d ", c[ i ][ j ] );
}
printf(" \n " );
}
getch( );
}
```

UNSOLVED PROGRAMS

1. Write a program to accept two matrixes and calculate the subtraction of two matrixes and display the result.

2. Write a program to accept a matrix and print the inverse of the matrix.

STRINGS

The way of a group of integers can be stored in an integer array, similarity a group of characters can be stored in a character array. Character arrays are many times also called strings.

A string constant is a one-dimensional array of characters terminated by a null (‘ \0 ’).

Example: char name[ ]={ ‘H’,’E’,’L’,’O’,’W’,’ \0’ }

Each character in the array occupies one byte of memory and the last character is always ‘ \0 ’.

Note: ‘ \0 ‘ is not necessary, C inserts the null character automatically.

24.Sample program of printing a string.

```
# include < stdio.h>
main ( )
{
char name [20]=” srinivasarao”;
int i=0;
while (i<=12)
{
printf( “% c”,name[ i ] );
i + +;
}
getch( );
}
```

(or)

```
main( )
{
char name[20]= “srinivasarao” ;
int i=0;
while(name[ i ]=’ \ 0’ )
{
printf (“ %c”,name[ i ] );
i++;
}
getch ( );
}
```

The %s is used in printf( ) is a format specification for printing out a string. The same specification can be used to receive a string from the keyboard, as show below program.

```
Main( )
{
char name[20];
printf( “enter your name “ );
scanf( “%s”,name);
printf( “%s”,name);
}
```

syntax: strcat(string1,string2);

ex: strcat(string 1, “ KKCC “);

```
main( )
{
char name[20],name1[20];
printf( “enter name”);
scanf(“%s”,name);
printf(“enter name1”);
scanf(“%s”,name1”);
printf (“the sum of two strings is : %s”,strcat(name,name1));
getch( );
}
```

strcpy( ):this function copies the contents of one string into another. The base addresses of the source and target strings should be supplied to this function.

Syntax: strlwr(string);

Ex: strlwr(“ KKCC”);

```
Main( )
{
char name[20],nname[20];
clrscr( );
printf(“enter the string”);
scanf(“%s”,name);
strcpy(nname,name);
printf(“%s”,name);
printf(“%s”,nname);
getch( );
}
```

strcmp( ):this function which compares two strings to find out whether they are same of different. If the two strings are identical, strcmp( ) returns a value zero. If they’re not, it returns the numeric difference between the ascii values of non-matching characters.

Syntax: strcmp(string1, string2);

Ex: strcmp(string1,”KKCC”);

```
Main( )
{
char name[20],name1[20];
int i;
printf("enter the name');
scanf("%s",name);
printf(" enter the name1");
scanf("%s",name1);
i=strcmp("name,name1");
if (i!=0)
{
printf ("the two strings are not same");
}
else
{
printf('the two strings are same');
}
getch( );
}
```

strrev( ):this function to reverse the string.

Syntax: strrev(string);

Ex: strrev("KKCC");

```
Main( )
{
char name[20]'
c;rscr( );
printf("enter the name");
scanf ("%s",name);
printf ("%s",strrev(name));
getch( );
}
```

GOTO STATEMENTS

This statement is used to transfer the control from one place to another place in the ' c ' program. That means this statement transfer the control to the specified label name. This statement is also called jumping statement or unconditional statement. The statement syntax is

```
Label:
…
…
Goto label;
Example:
A:
…
…
goto A;
```

25.Write a program to print multiple table of given number.

```
# include<stdio.h>
main( )
{
int n, i=1, k;
clrscr( );
printf( " enter the number ");
scanf("%d", &n);
A : k=n*i ;
Printf( "%d X %d = %d \n",n,i,k);
i= i+1;
if (i< =10)
{
goto A;
}
getch ( );
}
```

26.write a program to print 1 to 10 multiple tables

```
# include<stdio.h>
main ( )
{
int n=1,k,i;
clrscr( );
A: i=1;
B: k=n*i;
Printf( "%d X %d=%d \n",n,i,k);
i + +;
if(i< =10)
{
goto B;
}
n+ +;
if(n< =10)
{
goto A;
}
getch( );
}
```

27.Write a program to find out sum of positive integers the given value is 0 the program will be terminated.

```
# include<stdio.h>
main( )
{
int n,sum=0;
A: printf(" enter the number");
Scanf("%d",&n);
If (n< = 0)
{
goto B;
}
if (n>0)
{
sum=sum+n;
goto A;
}
B: printf( " the sum is %d ", sum);
getch ( );
}
```

28.write a program 1 to 10 multiple tables by leaving 6,9 integers.

```
# include <stdio.h>
main( )
{
int i,n=1,k;
clrscr( );
A: i=1;
B: k=n*i;
Printf(" %d X %d =%d\n",n,i,k);
i + +;
if (i< =10)
{
goto B;
}
C: n + +;
if(n= = 6 ||n = = 9)
{
goto C;
}
if(n< = 10)
{
goto A;
}
getch ( );
}
```

29.Write a program to print 1 to 15 multiple tables except 4 to 6,11 to 13.

```
# inclued <stdio.h>
main( )
{
int I,n=1,k;
A: i=1;
B; k=n*i;
Printf(" %d X %d=%d \n",n,i,k);
i+ +;
if (I< =10)
{
goto B;
}
C: n+ +;
If (n>3 && n<7)
{
goto C;
}
if (n= =11||n= =13)
{
goto C;
}
if (n< = 15)
{
```

```
goto A;
}
getch( );
}
```

SWITCH STATEMENT

A switch statement is a multiple branching statement using this statement a group of statements can be selected from several groups. The selection is based on the current value of the expression that is included with in the switch statement. The syntax of this statement is.

```
Switch (expression)
{
case(exp 1):
case(exp 2):
case(exp 3):
}
```

Where the expression evaluates to an integer value and it can also represents a character type. The statement is generally a compound statement,
Each group of statements must proceed with one or more case labels.

A group of statement can be label 'default'. This group will be selected if non of the case labels matches the value of the expression the default group may appear any where with in the switch statement.

Example:

```
Main( )
{
int i ;
printf ( “enter number’);
scanf (“ %d”,&i);
switch(i)
{
case 1:
printf(“one”);
case 2:
printf(“two”);
case 3:
printf(“three”);
default :
printf(“an error”);
break;
}
getch( );
}
```

BREAK: The break statement is used to terminate a loop or to exit from the switch statement. It can be used with in a while, do while, for and switch statements. The general form of the break statement is break;

CONTINUE: The continue statement is used to by pass some portion of the area in a loop does not terminate when the continue statement is encounter. The remaining portion of the loop as soon the continue statement executed is skipped. The continue statement can be included in while, do while and for statements the continue;

30.Write a program to read two numbers and does the following different operations through the menu.

1. Addition. 2. Subtraction. 3. Multiplication 4. Division.

```
# include<stdio.h>
main( )
{
int a,b,c,ch;
printf("enter two numbers");
scanf("%d%d",&a,&b);
clrscr( );
printf(" \t\t\t\t\t\tmenu\n");
printf(" \t\t\t\t\t\t 1.addition\n");
printf(" \t\t\t\t\t\t 2.subtraction\n");
printf(" \t\t\t\t\\t\t 3.multiplication\n");
printf(" \t\t\t\t\\t\t 4.devision\n");
printf("\n\nenter your choice");
scanf("%d",&ch);
switch(ch)
{
case 1:
c=a+b;
break;
case 2:
c=a-b;
break;
case 3:
c= a*b;
break;
case 4:
c=a/b;
break;
}
pirntf (“the result is =%d”,c);
getch( );
}
```

31.Write a program to read number of values in an array and perform the following operations through the menu by selecting choice.

1. Original order 2. Reverse order 3. sorted order
4. find maximum number 5. Find minimum number 6. Exit

```
# define MAX 20
main( )
{
int a[MAX],i,j,n,item,min,max;
printf(“enter how many numbers”);
scanf(“%d”,&n);
printf(“ enter numbers”);
for (i=0;i< n;i++)
{
scanf (“%d”,&a[i]);
}
clrscr( );
printf(“\t\t\t\t\t\tmenu\n”);
printf(“\t\t\t\t\t1.oringinal order\n”);
printf(“\t\t\t\t\t2.reverse order\n”);
printf(“\t\t\t\t\t3.sorted order\n”);
printf(“\t\t\t\t\t4.find maximum number\n”);
printf(“\t\t\t\t\t5.find minimum number\n”);
printf(“\t\t\t\t\t6.exit\n”);
printf(“\n\n\nenter your choice’);
scanf(“%d”,&ch);
switch(ch)
{
case 1:
{
printf(“the original order is “);
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
break;
}

case 2:
{
printf("the reverse order is ");
for(i=n-1;i>=0;i-)
{
printf("%d",a[i]);
}
break;
}

case 3:
{
for(i=0;i< n;i++)
for( j= i+1;j< n;j++)
{
if (a[i]> a[ j ])
{
int temp;
temp =a[i];
a[i]= a[ j ];
a[ j ]=temp.
}
}
}
printf "the sorted order is "  )
for (i=0;i<n;i++)
{
printf("%d",a[i]);
}
break;
}
case 4:
{
```

```
max=a[0];
for(j=1;j< n;j++)
{
if (a[ j ]> max)
max=a[ j ];
}
printf("the maximum value %d",max);
break;
}
case 5:
min=a[0];
for( j=1;j<n;j++)
{
if (a[ j ]>nin)
min=a[ j ];
}
printf("the minimum value %d",min);
break;
}
default :
printf("invalid option")
}
getch( );
}
```

FUNCTIONS

A function is a separate program segment that carries out some specific task or problem. Even 'c' program consists of one are more functions. One of these functions is main( ). A function will carry out it action when our it is process from some other portion of the program. Some function can be process from several different places with in the program. Once a function carries out its action, the control is transferred back to the calling portion of the program. A function return a single value information will be passed to the function using a special identifier called "arguments" or "parameters" and a result will be written to the calling program using the statement return some functions accept the information but do not return any value.

Defining a function: any function three parts those are

1. First line of the function gives the name of that function.
2. The argument declaration.

3. Body of the function.

The fist line of the function definition contains the type specification of the value return by the function followed by the function name. A set of arguments separated by commands and end closed in braces. The parameter list is optional. The type specification omitted if the function returns an i9nteger value. The general format of the first line of the function definition is

Syntax: data type name of the function
(Argument 1
Argument 2)

In the above syntax data types represents the data type return by the function value the argument1, argument2…are called "formal arguments" and these are also called "format parameters". The arguments allow information to be transferred from called portion of the program to the function.

The argument declaration follows the first line. Each for arguments must have the some data types as its corresponding actual arguments. The reminder of the function definition is body of the function, which will come after argument declaration the body of the function, is a compound statement. Information is return from the function to the calling portion of the program with the ' return"
Statement. This statement syntax is.

Syntax: return (expression…)

When a function not returning any value a simple return statement that is "return", can be sued to transfer the control from the function to the calling portion.

Processing a function: a function can be process by specifying its name followed by a list of arguments end closed with in the brackets and separated by commas. The argument appearing in the function is calling the actual argument for each formal argument. The actual arguments may express as constants, variable and expression each actual argument must be of some data type as its corresponding formal argument.

Sample programs:

Program to find the area of the rectangle by reading the values through the function.

```
# include< stdio.h>
 main( )
 {
 int l, w;
 printf (“Enter the length”);
 l=get( );
 printf (“enter the width’);
 w=get( );
 printf(“the area is :%d”,l*w);
 }
```

```
get ( )
{
int l;
scanf (“%d”,&l);
return( l );
}

# include<stdio.h>
int l,w,a;
main( )
{
printf(“enter length “);
l=get ( );
printf( “enter width”);
w=get ( );
c=calarea ( );
printf(“the area is “,c);
}
get( );
{
int l;
scanf(“%d”,&l);
return(l);
}
 calarea( );
{
 c=l *w;
```

Passing arguments to a function: when a single value to the function through the actual argument.The value of actual arguments is copied in to the function therefore the value of corresponding formal arguments can be modified with in the function, but the value of actual argument with in the calling program will not change this procedure for processing the value of arguments to a function is known as passing by value or call by value it allows on way communication from a calling portion input of the program to a function. Hence value parameters are also called input parameters to the function in such a type of parameters passing actual arguments can be constants, variables and expression.

Specifying argument data type: the calling program must contains forward declaration of the function being used by the program it is necessary if function return a non integer value and function called processed the function definition in forward declaration. It is possible to include the data type of the arguments with in the function, declaration this helps to detect the confused with in the data type

During the compilation to processing. The general form at forward declaration is

Syntax:- data type function name (argument 1, argument 2………)

Where data type represent the data type of the quantity return by the function. Argument 1,argument 2. Return to data type of arguments.

When function dose not return any value, then that type of the function can be specify as " void ".

Programs:

32.Write a program to calculate area of the rectangle using function with passing arguments.

```
Main( )
{
int l,w;
printf( "enter length" );
l=get( );
printf( "enter width");
w=get( );
printf( "\n the area is : %d",area(l,w));
}
get( )
{
int l;
scanf("%d",&l);
return( l );
}
area( x,y )
int x, y;
{
return(x * y);
}
```

33.Sample program using passing argument (with constant, variable and expression) to the function.

```
Main( )
{
int n=2;
printf(" passing constant value");
print(25);
printf(" passing the variables");
print(n);
printf(" passing the value of expression ");
print(n*2+1);
}
print(l);
intl;
{
printf("%d",l);
return( );
}
```

34.Write a program to find factorial of given number.

```
Main( )
{
int n,s;
rpintf("enter number")'
scanf("%d",&n);
s=get(n);
printf("the sum of the factorial nu8mber is %d",s);
}
```

Handling of non-integer functions:

By default the function returns a value of the type integer this is due to the appears of the type specified in the function header we must do two things to enable a calling function to receive a non integer value from a called function.

1. The type specified corresponding to the data type required must be mentioned in the function. The general form of the function definition is

Syntax: -

```
type specified function name (argument list)
Argument declaration.
{
Function statementation;
}
```

The type specified tells us the compiler the type of data the function is to return.

The called function must be declared at the start of the body in the calling function, like any other variables.

Ex:-

35. program to transfer floating number to the function

```
 Main( )
{
        float a, b,mul( );
        double div( );
        a=5.256;
        b=1.25;
        printf("\n %f",mul(a,b));
        printf("\n %u", div(a,b));
        }
        float mul (x,y);
        float x,y;
        {
        return(x*y);
        }
        double div(x,y);
        double x,y;
        {
        return(x/y);
        }
```

Functions with arrays: - like the values of simple variables it is also possible to pass the value of an array to a function. To pass an array to a called function, it is represent to list the name of the array with out any sub scripts and the size of the array as arguments. For example name (a,n); will pass all the elements containing the array 'a' of the size 'n'.

Ex :-

36. program to determined the largest value in the array through the function.

```
        Main( )
        {
int num[5]={50,10,30,40,20};
printf("%d", largest(num,5));
}
largest (x,y)
int x[5],y;
{
int max=x[0];
for( i=1; i< y; i++)
{
if (max < x[i])
{
max=x[i];
}
}
return(max);
}
```

37.Write a program that uses a function to sort an array of integers.

```
Main( )
{
int num[5]=(8,10,3,7,5),l;
printf ("the orginal numbers are :");
for ( l=0; l<5;;l++)
{
printf ("\n%d",nul[l]);
}
sort(num,5);
printf(" the sorted order is :);
for (l=0;l<5;l++)
{
printf("\n%de",num[l]);
}
}
sort(x,y);
int x[5],y;
{
int l,j,k;
for(l=0;l<y;l++)
{
for ( j=l+1; j< y; j++)
{
if (x[l]>x[i]
{
k=x[l];
x[l]=x[i];
x[ j ]= k ;
}
}
}
```

Storage Clauses :-_the storage class refers to refers to represent of variable and its score with in the program. That is the portion of the program over w3hich the variable is recognize also0 specified the location of variables there are for storage in the ' C ' there are

1. Automatic variable
2. External variable
3. Static variable
4. Register variable

1. **Automatic Variable: -**it is declared in side a function in which there are to be use there are created when the function is called and removed automatically when the function is exit automatic variable are therefor private or local to the function in which there are declared. Because of this property, automatic variable is also referring to as local or internal variables to define this kind of variable use 'auto' specified with declaration by default the 'C' compiler treated the variables as automatic.

38.Example program for automatic variable.

```
Main ( )
{
auto int a=1000;
fun2( );
printf("%d",a);
}
fun2( );
{
auto int a=10;
fun 1( );
printf("%d",a);
}
fun 1( );
{
auto int a=100;
printf("%d",a);
}
```

External Variable:-_ Variables that are declared out side a function are called external variables. These variables are also known as global variable.

39.Example program to external variable

```
Int x;
Main( )
{
x=10;
printf("%d\n",x);
printf ("x=%d\n",fun 1( ) );
printf ("x=%d\n",fun 2( ) );
printf ("x=%d\n",fun 3( ) );
}
fun 1( )
{
x=x+10;
return( x);
}
fun 2( )
{
int x;
x=1;
return( x );
}
fun 3( )
{
x=x+10;
return(x);
}
```

Static Variable: - The static variables may be either an internal type or external type depending on the place of declaration. Internal static variables are those which are declared inside a function. Therefor internal static variable are similar to auto variables except that they remind in through out the reminder of the program on external static variables are declared out side of all functions and is available to all the functions in that program. A variable can be declared as static using the keyword ‘ static ‘.

40.Example program for Static Variables.

```
Main( )
{
int l;
for(l=1; l< 3; l++)
{
stat( );
}
}
stat( )
{
static int x=0;
x=x+1;
printf("x=%d\n",x);
}
```

Registered Variables: -We can the compiler that a variable should be kept in one of the machine registers instead of keeping in the memory since a register variable process is much faster then a memory variable process. These kind of variable can be declared by using the keyword 'register'.

```
Ex: register int a;
    register  char ans;
```

Note: the most compilers allows only integer or character variables to be placed in the register.

```
#include <stdio.h>
#include <string.h>
struct tag {
char lname[20]; /* last name */
char fname[20]; /* first name */
int age; /* age */
float rate; /* e.g. 12.75 per hour */
};
struct tag my_struct; /* declare the structure my_struct */
int main(void)
{
strcpy(my_struct.lname,"Jensen");
strcpy(my_struct.fname,"Ted");
printf("\n%s ",my_struct.fname);
printf("%s\n",my_struct.lname);
return 0;
}
```

```
#include <stdio.h>
#include <string.h>
struct tag{ /* the structure type */
char lname[20]; /* last name */
char fname[20]; /* first name */
int age; /* age */
float rate; /* e.g. 12.75 per hour */
};
struct tag my_struct; /* define the structure */
void show_name(struct tag *p); /* function prototype */
int main(void)
{
struct tag *st_ptr; /* a pointer to a structure */
st_ptr = &my_struct; /* point the pointer to my_struct */
strcpy(my_struct.lname,"Jensen");
strcpy(my_struct.fname,"Ted");
printf("\n%s ",my_struct.fname);
printf("%s\n",my_struct.lname);
my_struct.age = 63;
show_name(st_ptr); /* pass the pointer */
return 0;
}
void show_name(struct tag *p)
{
printf("\n%s ", p->fname); /* p points to a structure */
printf("%s ", p->lname);
printf("%d\n", p->age);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define COLS 5
typedef int RowArray[COLS];
RowArray *rptr;
int main(void)
{
int nrows = 10;
int row, col;
rptr = malloc(nrows * COLS * sizeof(int));
for (row = 0; row < nrows; row++)
{
for (col = 0; col < COLS; col++)
{
rptr[row][col] = 17;
}
}
return 0;
}
```