ARRAYS AND STRINGS

An array is a collection of variables of the same type that are referenced by a common name. A specific element in an array is accessed by an index.

ONE DIMENSIONAL ARRAYS:

The general form for declaring a single – dimensioned array is

```
type variablename [ size ] ;
```

*** Arrays must be explicitly declared so that the compiler may allocate space for them in memory.**

**** Here, type declares the type of the array.**

***** size defines how many elements the array will hold.**

For example

```
float balance[100];
```

For example, the following program loads an integer array with the numbers 0 through 99:

```
Void main (void)
{
        int x[100];
        int i, j ;
        for ( i = 0 ; i < 100 ; ++i)
        x [i ] =i;
        for (j=0;j<100;j++)
        printf ( " %d \n", x[j]);
        return 0 ;
}
/ * Initializing an array * /
# include " stdio.h"

main ( )
{
        int n[10], i ;
        for ( i = 0 ; i < = 9 ; i ++ )
                n[i] =1 0 ;
        printf(" %s %13s \n", " Element ", " value " );
        for ( i = 0 ; i < = 9 ; i++)
                printf ( " %7d %13d \n", i, n[i] ) ;
        return 0;
}
/* Initializing an array with a declaration * /

# include " stdio.h"
main ()
{
        int i,  n[10] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

        printf ( " %s % 13s \n ", " Element", " value" );

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

                printf( " %7d %13 d \n ", i, n[i] ) ;
        return 0 ;
}
```

```
/* Initialize the elements of array s to the even integers from 2 to 20 * /
#include “stdio.h”
#define size 10
main()
{
        int s[size], j;
        for ( j = 0 ; j < = size – 1 ; j ++ )
                s[j] = 2 + 2 * j ;
        printf( “ %s %13s \n “, “ Element “, “ Value “ ) ;
                for ( j = 0 ; j < = size – 1 ; j ++ )
printf ( “ %7d %13d \n “, j, s[j] );
return 0;
}
```

```
/* Compute the sum of the elements of the array */
# include "stdio.h"
#define size 12
main()
{
        int a[size] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 } ;
        int i, sum = 0;

        for ( i = 0 ; i < = size – 1; i ++ )
                sum + = a[i];

        printf ( " sum  of array element values is %d \n", sum) ;
        return 0;
}
```

Histogram Printing Using array

```
# include "stdio.h"
#define size 10
main ()
{
        int n[size] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
        int i, j ;
        printf(" %s %13s %17s \n", " Element", " value", " Histogram" ) ;

        for ( i = 0; i < = size – 1 ; i ++)

                {
                        printf( " %7d %13d            " , i, n[i] );

                                for ( j = 1; j<= n[i]; j++)
                                printf ( " %c", '*');
                        printf(“ \n ”);
                }
        return 0 ;
}

/* Roll a six – sided die 6000 times * /
#include “stdio.h”
#include “stdlib.h”
#include “time.h”
#define size 7

main ()
{
        int face, roll, frequency [ size ] = {0};
        srand( time (NULL));
        for ( roll = 1; roll < = 6000 ; roll ++ )

                {
                        face = rand( ) % 6 + 1;

                        ++ frequency [ face ] ;
                }
        printf ( “ %s %17 s \n “, “ Face”, “ Frequency “ );

        for ( face = 1; face < = size – 1 ; face++)
                printf(“ % 4d %17d \n”, face, frequency[face] );
        return 0;
}
```

```
/* Treating Character arrays as strings */

# include "stdio.h"
main ()
{
        char string1[20], string2[ ] = "string literal" ;
        int i ;

        printf ( "Enter a string :" ) ;
        scanf ( "%s", string1);
        printf(" string1 is : %s \n string2: is  %s  \n"
                " string1 with spaces between characters is : \n",   string1, string2);

        for ( i = 0 ; string1[i] ! = ' \ 0 ' ; i++)
                printf("%c", string1[i]);
        printf( " \n" );
        return 0;
}
/* Sorting arrays values into ascending order */

# include "stdio.h"
# define size 10

main ()
{
        int a[size] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

        int  i, j, temp ;

        printf ( " Data items in given order \n" );
        for ( i = 0 ; i < = size – 1 ; i ++ )
        printf ( " % 4 d ", a[i] ) ;

        for ( j = 1 ; i < = size – 1 ; j ++ )

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

                        if ( a[i] > a[i+1] )

                        {       temp = a[i];
                                a[i] = a[i+1];
                                a[i+1] = temp ;
                        }

                printf ( " \n Data items is ascending order \n " );

                for ( i = 0 ; i < = size – 1 ; i ++ )

                printf ( " %4d", a[i] );

                printf ( " \n" );

                return 0;
}
```

Additional Notes – For Reference

WHAT IS A POINTER VARIABLE IN C-LANGUAGE?

A Pointer is variable that holds address of a memory. This address locates the address of another variable in memory. If one variable contains the address of another variable, the first variable is said to point to the second.

The Syntax of the Pointer variable is

data_type   *name_of_variable;

The Pointer Operators:

There are two special pointer operators**: * and &.**

1. The & is a unary operator that returns the memory address of its operand.

For example,

```
m = &count;
```

places into m the memory address of the variable count. This address is the computers internal location of the variable.

2. The pointer operator * is the complement of &. It is a unary operator that returns the value of the variable located at the address.

For example, if m contains the memory address of the variable count,

```
q= *m;
```

If the value at the memory address of m is 200, then q = 200.

Check Whether the value of x is assigned to y ?

```
void main (void)
{
        float x, y ;
        int *p ;
        p = &x;
        y = *p;
}
```

Here the x is not assigned to y, because p is declared as an integer pointer, only 2 bytes of information will be transferred to y, not the 8 bytes that normally make up a floating-point number.

Pointer Assignments

As like any variable, you may use a pointer in the RHS of assignment statements to assign its value to another pointer.
For example,

```
# include " stdio.h"

void main (void)
{
        int x ;
        int *p1, *p2;
        p1 = &x;
        p2 = p1;
        printf(" %p", p2) ;     /* Print the address of x, not x's value ! */
}
```

Both p1 and p2 now point to x. The address of x is displayed by using the %p printf() format modifier, which causes printf() to display an address in the format used by the host computer.

```
/* Cube a variable using call by value */
#include "stdio.h"
int cubebyvalue(int);
main()
{
        int number = 5;

        printf (" The original value of number is %d \n", number);

        number = cubebyvalue(number);

        printf(" The new value of the number is %d \n", number);

        return 0;

}
int cubebyvalue( int n)
{
        return n*n*n;                   /*      Cube local variable n  */
}

/* Cube a variable using call by reference */

#include "stdio.h"
```

```
void cubebyreference(int *);

main()

{
        int number = 5;

        printf(" The original value of number is %d \n", number);

        cubebyreference(&number);

        printf(" The new value of number is %d \n", number);

        return 0;
}
void cubebyreference ( int *nptr)
{
        *nptr = *nptr * *nptr * *nptr;          /* cube number in main */
}



/* Converting lowercase letters to uppercase letters */
/* using a non-constant pointer to non-constant data */

#include "stdio.h"

void convertToUppercase(char *);

main()
{
        char string[] = “characters”;

        printf(“ The string before conversion is: %s \n”, string);

        convertToUppercase(string);

        printf(“ The string after conversion is: %s\n”, string);

        return 0:
}
void convertToUppercase(char *s)
{
        while (*s ! = ‘\0’)
{
        if (*s >= ‘a’ && *s <= ‘z’ )

        *s - = 32;      /* convert to ASCII uppercase letter */

        ++s;            /* increment s to point to the next character */
}

}
```

```
/* Printing a string one character at a time using a non-constant pointer to constant data */

#include “stdio.h”

void printcharacters(const char *);
main()
```

```
{
        char string[] = “print characters of string “;

        printf(“The string is : \n”);

        printcharacters(string);

        putchar(‘\n’);

        return 0;
}
void printcharacters(const char *s)
{
        for( ;  *s ! = ‘\0’ ; s++ )          /* No initialization */

                putchar(*s);
}

/* Attempting to modify data through a non-constant pointer to constant data */

#include “stdio.h”

void f(const int *);

main()
{
        int y=10;
        f(&y);            /* f attempts illegal modification */

        return 0;
}
```

```
void f(const int *x)
{
        *x = 100 ;      /* cannot modify a const object */  /* Compile error */
}
```

/* Attempting to modify a constant pointer to non-constant data */

```
#include "stdio.h"
main()

{
        int x,y;
        int * const ptr = &x;

        ptr = &y ;

        return 0;               /* Compile Error */

}
```

/* This program puts values into an array, sorts the values into ascending order, and prints the resulting array */

```
#include "stdio.h"
#define size 10
void bubblesort(int *, int);

main()
{
        int i, a[size] = { 2,6,4,8,10,12,89,68,45,37};
        printf(“ Data items in original order \n”);

        for ( i = 0 ; i < = size – 1 ; i++)

                printf(“%4d”, a[i]);

        bubblesort(a, size);
        printf(“ \n Data items in ascending order \n”);

        for ( I = 0 ; i<= size – 1 ; i++)
        printf(“%4d”, a[i]);
        printf(“\n”);
        return 0;
}
void bubblesort(int *array, int size)
{
        int pass, j;

        void swap(int *, int *);

        for (pass = 1; pass <=size – 1 ; pass++)

        for (j=0;j<=size – 2; j++)

                if (array[j] > array[j+1])

                swap(&array[j], &array[j+1]);
}

void swap(int *element1ptr, int *element2ptr)
```

```
{
        int temp;
        temp = *element1ptr;
        *element1ptr = *element2ptr;
        *element2ptr = temp;

}
```

Examples

Mean and standard deviation

```
//* C program to find the value of mean and standard deviation*//
#include<studio.h>
#include<conio.h>
#include<math.h>
#define max 10
void main()
{
inti,n=0;
float x[max], deviation,sum,sumsqr,mean, variance,sd;
sum=sumsqr=0;
clrscr();
printf("\n input values: int -1 to end");
for(i=0;i<max;i++)
{
scanf("%f",&x[i]);
if(x[i]==-1)
break;
sum+=x[i];
n+=1;
}
mean=sum/(float) n;
for (i=1;i<n;i++)
{
```

```
deviation=x[i]-mean;
sumsqr+=deviation*deviation;
}
variance=sumsqr/(float) n;
sd=sqrt(variance);
printf("\n number of items :%d \n",n);
printf("\n mean =%f",mean);
printf("\nstandard deviation=%f \n",sd);
}
```

Correlation Program

```
//*C program to calculate correlation co-efficient
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main ()
{
inti,n;
float x[50],y[50],sumx,Sumy,meanx,meany,sum_sx,sum_sy,sum_xy,r;
clrscr();
sumx=sumy=sum_sx=sum_sy=0;
printf ("\n enter the n value….");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf ("\n enter the x,y value….\n");
scanf("%f,%f",&x[i],&y[j]);
sumx=sumx+x[i];
sumy=sumy+y[i];
}
meanx=sumx/n;
meany=sumy/n;
for(i=0;i<n;i++)
{
```

```
sum_sx=sum_sx+(x[i]-meanx)*(x[i]-meanx);
sum_sy=sum_sy+(y[i]-meany)*(y[i]-meany);
sum_xy=sum_xy+(x[i]-meanx)*(y[i]-meany);
}
r=sum_xy/sqrt(sum_sx*sum_sy);
printf (“\n the correlation coefficient :r=%f”,r);
getch();
}
```