Arrays

```
main( )
{
      int  x ;
      x = 5 ;
      x = 10 ;
      printf ( "\nx = %d", x ) ;
}
```

- This program will print the value of **x** as 10. Why so?
- Because when a value 10 is assigned to **x**, the earlier value of **x**, i.e. 5, is lost.
- Thus, ordinary variables (the ones which we have used so far) are capable of holding only one value at a time (as in the above example).
- However, there are situations in which we would want to store more than one value at a time in a single variable.

- Suppose we wish to arrange the percentage marks obtained by 100 students in ascending order. In such a case we have two options to store these marks in memory:

(a) Construct 100 variables to store percentage marks obtained by 100 different students, i.e. each variable containing one student's marks.

(b)Construct one variable (called array or subscripted variable) capable of storing or holding all the hundred values.

- The second alternative is better.
- A simple reason for this is, it would be much easier to handle one variable than handling 100 different variables.
- int a1,a2,a3,a4,……..,a100;
- Int a[100];

Array

- An array is a collective name given to a group of ‘similar quantities’.
- These similar quantities could be percentage marks of 100 students, or salaries of 300 employees, or ages of 50 employees.
- What is important is that the quantities must be ‘similar’.
- Each member in the group is referred to by its position in the group

- assume the following group of numbers, which represent percentage marks obtained by five students.

  per = { 48, 88, 34, 23, 96 }
- If we want to refer to the second number of the group, the usual notation used is per 2. Similarly, the fourth number of the group is referred as per 4.
- However, in C, the fourth number is referred as **per[3].**
- This is because in C the counting of elements begins with 0 and not with 1.
- Thus, in this example **per[3]** refers to 23 and
- **per[4]** refers to 96.
- In general, the notation would be **per[i]**, where, **i** can take a value 0, 1, 2, 3, or 4, depending on the position of the element being referred.
- Here **per** is the subscripted variable (array), whereas **i** is its subscript.

- An array is a collection of similar elements.
- These similar elements could be all **int**s, or all **float**s, or all **char**s, etc.
- Usually, the array of characters is called a ‘string’, whereas an array of **int**s or **float**s is called simply an array.

Program to find average marks obtained by a class of 30 students in a test.

```
main( )
{
 int avg, sum = 0 ;
int i ;
int marks[30] ; /* array declaration */
for ( i = 0 ; i <= 29 ; i++ )
{
 printf ( "\nEnter marks " ) ;
scanf ( "%d", &marks[i] ) ; /* store data in array */
}
for ( i = 0 ; i <= 29 ; i++ )
sum = sum + marks[i] ; /* read data from an array*/
avg = sum / 30 ;
printf ( "\nAverage marks = %d", avg ) ;
}
```

```
main( )
{
 int avg, sum = 0 ;
int i , marks;
 printf ( "\nEnter marks " ) ;
scanf ( "%d", &marks ) ;
sum = sum + marks;
printf ( "\nAverage marks =
%d", avg ) ;
```

Array Declaration

```
int marks[30] ;
```

- Here, **int** specifies the type of the variable, just as it does with ordinary variables and the word **marks** specifies the name of the variable.
- The **[30]** however is new.
- The number 30 tells how many elements of the type **int** will be in our array.
- This number is often called the ‘dimension’ of the array.
- The bracket ([ ]) tells the compiler that we are dealing with an array

Accessing Elements of an Array

- This is done with subscript, the number in the brackets following the array name.
- This number specifies the element's position in the array.
- All the array elements are numbered, starting with 0.
- Thus, **marks[2]** is not the second element of the array, but the third.
- In our program we are using the variable **i** as a subscript to refer to various elements of the array.
- This variable can take different values and hence can refer to the different elements in the array in turn.

Entering Data into an Array

```
for ( i = 0 ; i <= 29 ; i++ )
{
   printf ( "\nEnter marks " ) ;
   scanf ( "%d", &marks[i] ) ;
}
```

- The **for** loop causes the process of asking for and receiving a student's marks from the user to be repeated 30 times.
- The first time through the loop, **i** has a value 0, so the **scanf( )** function will cause the value typed to be stored in the array element **marks[0],** the first element of the array.
- This process will be repeated until **i** becomes 29.
- There is no array element like **marks[30]**.
- In **scanf( )** function, we have used the "address of" operator (&) on the element **marks[i]** of the array, just as we have used it earlier on other variables **(&rate**, for example).

Reading Data from an Array

```
for ( i = 0 ; i <= 29 ; i++ )
        sum = sum + marks[i] ;
```

Review

- An array is a collection of similar elements.
- The first element in the array is numbered 0, so the last element is 1 less than the size of the array.
- An array is also known as a subscripted variable.
- Before using an array its type and dimension must be declared.
- However big an array its elements are always stored in contiguous memory locations.

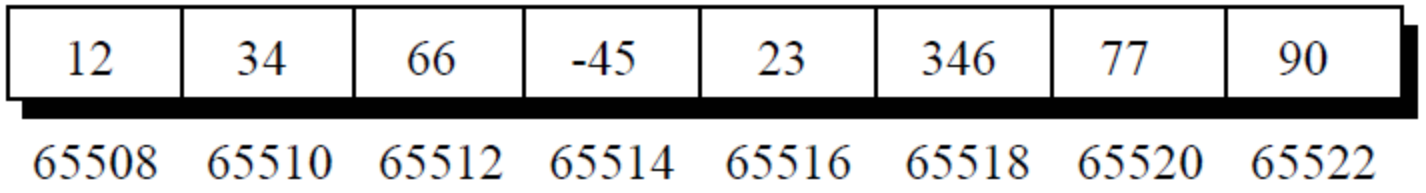
Array Initialisation

- int num[6] = { 2, 4, 12, 5, 45, 5 } ;
- int n[ ] = { 2, 4, 12, 5, 45, 5 } ;
- float press[ ] = { 12.3, 34.2 -23.4, -11.3 } ;

- Till the array elements are not given any specific values, they are supposed to contain garbage values.
- If the array is initialized where it is declared, mentioning the dimension of the array is optional as in the 2nd example above.

What happens in memory when we make this declaration?

- int arr[8] ;
- 16 bytes get immediately reserved in memory.
- 2 bytes each for the 8 integers
- Since the array is not being initialized, all eight values present in it would be garbage values



Bounds Checking

- In C there is no check to see if the subscript used for an array exceeds the size of the array.
- Data entered with a subscript exceeding the array size will simply be placed in memory outside the array; probably on top of other data, or on the program itself.
- This will lead to unpredictable results, and there will be no error message to warn you that you are going beyond the array size.

```
main( )
{
int num[40], i ;
for ( i = 0 ; i <= 100 ; i++ )
 num[i] = i ;
}
```

Program 1

Five numbers are entered from the keyboard into an array.

The number to be searched is entered through the keyboard by the user.

Write a program to find if the number to be searched is present in the array

and if it is present, display the number of times it appears in the array.

```c
#include<stdio.h>
int main()
{
    int a[5] , i, count=0,num;
    for(i=0;i<5;i++)
    {
    printf("Enter a[%d]", i+1);
    scanf("%d", &a[i]);

    }
```

Entering Data in the array

```c
#include<stdio.h>
int main()
{
    int a[5] , i, count=0,num;
    for(i=0;i<5;i++)
    {
    printf("Enter a[%d]", i+1);
    scanf("%d", &a[i]);

    }
    printf("Enter the number you want to search");
    scanf("%d", &num);
```

Entering Data in the array

```c
#include<stdio.h>
int main()
{
    int a[5] , i, count=0,num;
    for(i=0;i<5;i++)
    {
    printf("Enter a[%d]", i+1);
    scanf("%d", &a[i]);

    }
    printf("Enter the number you want to search");
    scanf("%d", &num);
    for(i=0;i<5;i++)
    {
        if(a[i]==num)
        count++;
    }
    printf("%d has occured %d times", num,count);
    return 0;
}
```

Entering Data in the array

Comparing num with every element of array

```
Enter a[1]2
Enter a[2]3
Enter a[3]4
Enter a[4]2
Enter a[5]2
Enter the number you want to search2
2 has occured 3 times
```

Program 2

- Ten numbers are entered from the keyboard into an array.
- Write a program to find out how many of them are positive, how many are negative, how many are even and how many odd.

```c
int main()
{
    int a[10] , i, countp=0,counto=0, countn=0;
    for(i=0;i<10;i++)
    {
    printf("Enter a[%d]", i+1);
    scanf("%d", &a[i]);

    }
    for(i=0;i<10;i++)
    {
        if(a[i]>0)
        countp++;
        else if(a[i]<0)
        countn++;
        else
        counto++;
    }
    printf("\nNo of positive number is %d", countp);
    printf("\nNo of negative number is %d", countn);
    printf("\nNo of zero number is %d", counto);
    return 0;
}
```

```
Enter a[1]-2
Enter a[2]-2
Enter a[3]-3
Enter a[4]-4
Enter a[5]0
Enter a[6]0
Enter a[7]0
Enter a[8]0
Enter a[9]2
Enter a[10]3

No of positive number is 2
No of negative number is 4
No of zero number is 4
```

Ex1: Write a program to find largest and smallest element in an array

Algorithm

1. Input array a[10], max and min.
2. Initialize max =a[0], min=a[0]
3. Set i = 0
4. Repeat steps 5 to 7 while i<10
5. if(a[i]>max) then max= a[i]
6. if(a[i]<min) then min=a[i]
7. Increment i

by 1 and go to step 4

9. Print max and min
10. End

Ex 1: Write a program to find largest and smallest element in an array

```c
#include<stdio.h>
int main()
{
    int a[10], i, max, min;
```

Array declaration

Ex 1: Write a program to find largest and smallest element in an array

```c
#include<stdio.h>
int main()
{
    int a[10], i, max, min;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d",i);
        scanf("%d",&a[i]);
    }
```

Assigning values to the array elements

Ex 1: Write a program to find largest and smallest element in an array

```c
#include<stdio.h>
int main()
{
    int a[10], i, max, min;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d",i);
        scanf("%d",&a[i]);
    }
    max=min=a[0];
    for(i=1;i<10;i++)
    {
        if(a[i]>max)
        max=a[i];
        if(a[i]<min)
        min=a[i];
    }
    printf("maximu number = %d and minimum number=%d", max, min);
}
```

Comparing max and min with every element of the array

Ex 2: Write a program to increment the array elements by 1

```c
#include<stdio.h>
int main()
{
    int a[10], i;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d]",i);
        scanf("%d",&a[i]);
    }
```

Ex 2: Write a program to increment the array elements by 1

```c
#include<stdio.h>
int main()
{
    int a[10], i;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d]",i);
        scanf("%d",&a[i]);
    }

    for(i=0;i<10;i++)
    {
        a[i] = a[i]+1;
    }
```

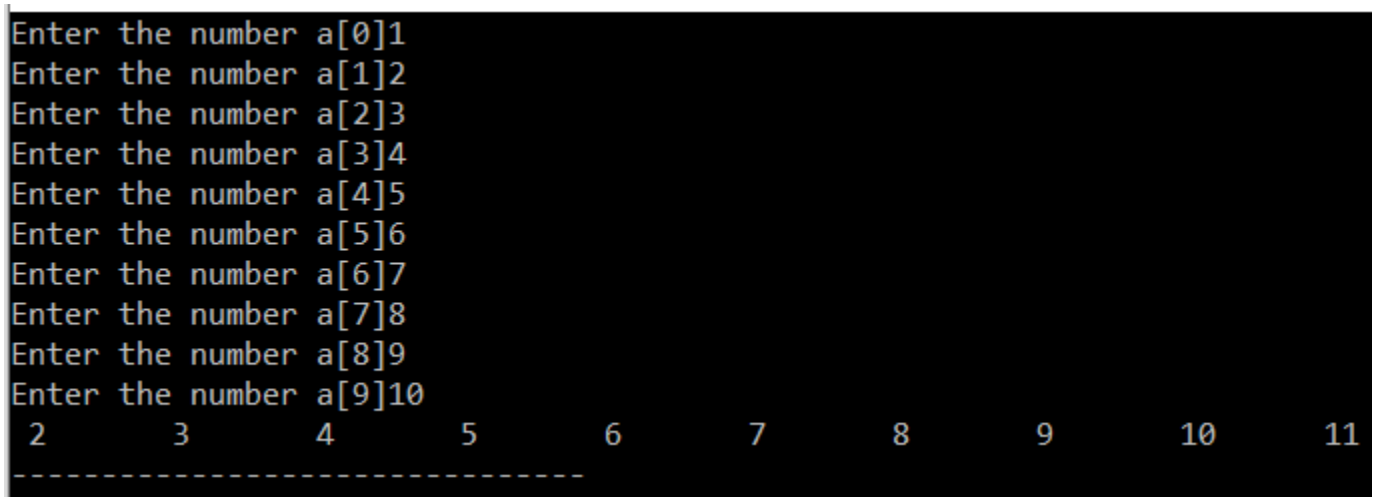
Ex 2: Write a program to increment the array elements by 1

```c
#include<stdio.h>
int main()
{
    int a[10], i;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d]",i);
        scanf("%d",&a[i]);
    }

    for(i=0;i<10;i++)
    {
        a[i] = a[i]+1;
    }

    for(i=0;i<10;i++)
    {
        printf("%d",a[i]);
    }
}
```

Ex 2: Write a program to increment the array elements by 1



Ex 2: Write a program to increment the array elements by 1

```c
#include<stdio.h>
int main()
{
    int a[10], i;
    for(i=0;i<10;i++)
    {
        printf("Enter the number a[%d]",i);
        scanf("%d",&a[i]);
    }

    for(i=0;i<10;i++)
    {
        a[i] = a[i]+1;
    }

    for(i=0;i<10;i++)
    {
        printf("a[%d]= %d \t",i, a[i]);
    }
}
```

Ex 2: Write a program to increment the array elements by 1

```
Enter the number a[0]1
Enter the number a[1]2
Enter the number a[2]3
Enter the number a[3]4
Enter the number a[4]5
Enter the number a[5]6
Enter the number a[6]7
Enter the number a[7]8
Enter the number a[8]9
Enter the number a[9]10
a[0]= 2         a[1]= 3         a[2]= 4         a[3]= 5         a[4]= 6         a[5]= 7         a[6]= 8         a[7]= 9
        a[8]= 10        a[9]= 11
--------------------------------
```

Exercise 3

- Write a program to calculate average marks of 5 students in 3 subjects English, hindi, maths and .
- Print the average marks of each student along with his roll no

- How many arrays required?

- How many arrays required?
- Eng[5], hindi[5], maths[5], avg[5], rollno[5]

- How to enter elements of each array?

- How to enter elements of each array?
- Using for loop

Exercise 3

```c
#include<stdio.h>
int main()
{
    int avg[5], i, eng[5], hindi[5], maths[5], rollno[5];
    for(i=0;i<5;i++)
    {
        printf("Enter the roll number rollno[%d]",i+1);
        scanf("%d",&rollno[i]);

        printf("Enter the marks in english[%d]",i+1);
        scanf("%d",&eng[i]);

        printf("Enter the marks in hindi[%d]",i+1);
        scanf("%d",&hindi[i]);

        printf("Enter the marks in maths[%d]",i+1);
        scanf("%d",&maths[i]);
    }
```

Assigning values to the array elements

- How to calculate average?

- How to calculate average?
- Using loops

Exercise 3

```c
#include<stdio.h>
int main()
{
    int avg[5], i, eng[5], hindi[5], maths[5], rollno[5];
    for(i=0;i<5;i++)
    {
        printf("Enter the roll number rollno[%d]",i+1);
        scanf("%d",&rollno[i]);

        printf("Enter the marks in english[%d]",i+1);
        scanf("%d",&eng[i]);

        printf("Enter the marks in hindi[%d]",i+1);
        scanf("%d",&hindi[i]);

        printf("Enter the marks in maths[%d]",i+1);
        scanf("%d",&maths[i]);
    }

    for(i=0;i<5;i++)
    {
        avg[i] = (eng[i]+hindi[i] + maths[i])/3;
    }
```

```c
#include<stdio.h>
int main()
{
    int avg[5], i, eng[5], hindi[5], maths[5], rollno[5];
    for(i=0;i<5;i++)
    {
        printf("Enter the roll number rollno[%d]",i+1);
        scanf("%d",&rollno[i]);

        printf("Enter the marks in english[%d]",i+1);
        scanf("%d",&eng[i]);

        printf("Enter the marks in hindi[%d]",i+1);
        scanf("%d",&hindi[i]);

        printf("Enter the marks in maths[%d]",i+1);
        scanf("%d",&maths[i]);
    }

    for(i=0;i<5;i++)
    {
        avg[i] = (eng[i]+hindi[i] + maths[i])/3;
    }

   for(i=0;i<5;i++)
   {
       printf(" rollno[%d] has secured  %d as average marks\n", rollno[i],avg[i]);
   }
}
```

Exercise 3

```
Enter the roll number rollno[1]12
Enter the marks in english[1]30
Enter the marks in hindi[1]20
Enter the marks in maths[1]40
Enter the roll number rollno[2]23
Enter the marks in english[2]30
Enter the marks in hindi[2]40
Enter the marks in maths[2]50
Enter the roll number rollno[3]45
Enter the marks in english[3]50
Enter the marks in hindi[3]56
Enter the marks in maths[3]65
Enter the roll number rollno[4]13
Enter the marks in english[4]23
Enter the marks in hindi[4]34
Enter the marks in maths[4]45
Enter the roll number rollno[5]3
Enter the marks in english[5]30
Enter the marks in hindi[5]45
Enter the marks in maths[5]56
 rollno[12] has secured  30 as average marks
 rollno[23] has secured  40 as average marks
 rollno[45] has secured  57 as average marks
 rollno[13] has secured  34 as average marks
 rollno[3] has secured  43 as average marks
```

Accessing Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr[] = {51,32,43,24,5,26};
int i;
for(i=0; i<=5; i++) {
printf("\n%d %d %d %d",arr[i],*(i+arr),*(arr+i),i[arr]);
}
getch();
}
```

Accessing Array

```
#include<stdio.h>
#include<conio.h>
void main()
{
int arr[] = {51,32,43,24,5,26};
int i;
for(i=0; i<=5; i++) {
printf("\n%d %d %d %d",arr[i],*(i+arr),*(arr+i),i[arr]);
}
getch();
}
```

```
Output:
51 51 51 51
32 32 32 32
43 43 43 43
24 24 24 24
5 5 5 5
26 26 26 26
```

Operations with One Dimensional Array

1. Deletion – Involves deleting specified elements form an array.
2. Insertion – Used to insert an element at a specified position in an array.
3. Searching – An array element can be searched. The process of seeking specific elements in an array is called searching.
4. Merging – The elements of two arrays are merged into a single one.
5. Sorting – Arranging elements in a specific order either in ascending or in descending order.

C Program for deletion of an element from the specified location from an Array

```
#include<stdio.h>
int main() {
int arr[30], num, i, loc;
printf("\nEnter no of elements:");
scanf("%d", &num);
//Read elements in an array
printf("\nEnter %d elements :", num);
for (i = 0; i < num; i++) {
scanf("%d", &arr[i]);    }
//Read the location
printf("\nLocation of the element to be deleted :");
scanf("%d", &loc);
/* loop for the deletion  */
while (loc < num) {
arr[loc - 1] = arr[loc];
loc++;    }
num--;  // No of elements reduced by 1
//Print Array
for (i = 0; i < num; i++)
printf("\n %d", arr[i]);
return (0);
}
```

C Program for deletion of an element from the specified location from an Array

```
#include<stdio.h>
int main() {
int arr[30], num, i, loc;
printf("\nEnter no of elements:");
scanf("%d", &num);
//Read elements in an array
printf("\nEnter %d elements :", num);
for (i = 0; i < num; i++) {
scanf("%d", &arr[i]);    }
//Read the location
printf("\nLocation of the element to be deleted :");
scanf("%d", &loc);
/* loop for the deletion  */
while (loc < num) {
arr[loc - 1] = arr[loc];
loc++;    }
num--;  // No of elements reduced by 1
//Print Array
for (i = 0; i < num; i++)
printf("\n %d", arr[i]);
return (0);
}
```

0	1	2	3	4	5	6	7	8
22	11	10	6	8	12	16	4	6

C Program for deletion of an element from the specified location from an Array

```
#include<stdio.h>
int main() {
int arr[30], num, i, loc;
printf("\nEnter no of elements:");
scanf("%d", &num);
//Read elements in an array
printf("\nEnter %d elements :", num);
for (i = 0; i < num; i++) {
scanf("%d", &arr[i]);    }
//Read the location
printf("\nLocation of the element to be deleted :");
scanf("%d", &loc);
/* loop for the deletion  */
while (loc < num) {
arr[loc - 1] = arr[loc];
loc++;    }
num--;  // No of elements reduced by 1
//Print Array
for (i = 0; i < num; i++)
printf("\n %d", arr[i]);
return (0);
}
```

0	1	2	3	4	5	6	7	8
22	11	6	8	12	4	16	6	6

Loc =3
A[2] = a[3]
A[3]=a[4]
A[4]=A[5]
NUM--

Exercise

- Write a program to delete multiple occurrences of a number

0	1	2	3	4	5	6	7	8	9
2	2	2	2	2	5	2	2	2	2

0	1	2	3	4	5	6			
4	1	4	1	2	5	6			

C Program to insert an element in an array

```
int arr[30], element, num, i, location;
printf("\nEnter no of elements:");
scanf("%d", &num);
for (i = 0; i < num; i++) {
scanf("%d", &arr[i]); }
printf("\nEnter the element to be inserted:");
scanf("%d", &element);
printf("\nEnter the location");
scanf("%d", &location);
//Create space at the specified location
for (i = num; i >= location; i--) {
arr[i] = arr[i - 1];   }
num++;
arr[location - 1] = element;
//Print out the result of insertion
for (i = 0; i < num; i++)
printf("n %d", arr[i]);
return (0);
}
```

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]
6	5	4	3	2	1

Num=6, loc =2	1,2,3,4,5,6 Element =99
Arr[6] =ARR[5]	1
Arr[5]=ARR[4]	2
Arr[4]=ARR[3]	3
Arr[3]=ARR[2]	4
Arr[2]=ARR[1]	5

Arr[0]	Arr[1]	Arr[2]	Arr[3]	Arr[4]	Arr[5]	Arr[6]
6	99	5	4	3	2	1

Exercise

- Write a program to search for an element in an array.
- Write a program to copy contents of one array into another
- Write a program to insert an element in a sorted list

0	1	2	3	4	5	6	7
2	4	6	8	10	12	14	16

C program to merge two arrays in C Programming and create a sorted list

```
#include<stdio.h>
int main() {
int arr1[30], arr2[30], res[60];
int i, j, k, n1, n2;
printf("\nEnter no of elements in 1st array:");
scanf("%d", &n1);
for (i = 0; i < n1; i++) {
scanf("%d", &arr1[i]); }
printf("\nEnter no of elements in 2nd array:");
scanf("%d", &n2);
for (i = 0; i < n2; i++) {
scanf("%d", &arr2[i]); }
i = 0;
j = 0;
k = 0;
```

C program to merge two arrays in C Programming and create a sorted list

```
// Merging starts
while (i < n1 && j < n2) {
if (arr1[i] <= arr2[j]) {
res[k] = arr1[i];
i++;
k++; }
else {
res[k] = arr2[j];
k++;
j++; }
}
/*Some elements in array 'arr1' are still remaining where as the array
'arr2' is exhausted*/
while (i < n1) {
res[k] = arr1[i];
i++;
k++; }
```

C Program to display array elements with addresses

```
#include<stdio.h>
#include<stdlib.h>
#define size 10
int main() {
int a[3] = { 11, 22, 33 };
printf("\n a[0],value=%d : address=%u", a[0], &a[0]);
printf("\n a[1],value=%d : address=%u", a[1], &a[1]);
printf("\n a[2],value=%d : address=%u", a[2], &a[2]);
return (0);
}
```

C Program to display array elements with addresses

```
#include<stdio.h>
#include<stdlib.h>
#define size 10
int main() {
int a[3] = { 11, 22, 33 };
printf("\n a[0],value=%d : address=%u", a[0], &a[0]);
printf("\n a[1],value=%d : address=%u", a[1], &a[1]);
printf("\n a[2],value=%d : address=%u", a[2], &a[2]);
return (0);
}
```

Output:

```
a[0],value=11 : address=2358832
a[1],value=22 : address=2358836
a[2],value=33 : address=2358840
```

Strings

- The way a group of integers can be stored in an integer array, similarly a group of characters can be stored in a character array.
- Character arrays are many a time also called strings.
- Character arrays or strings are used by programming languages to manipulate text such as words and sentences.

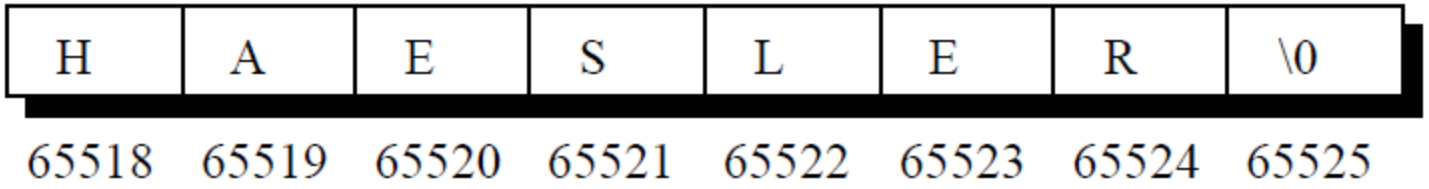
- A string constant is a one-dimensional array of characters terminated by a null (‘\0’).
- For example,

```
char name[ ] = { 'H', 'A', 'E', 'S', 'L', 'E', 'R', '\0' } ;
```

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

NULL character

- It looks like two characters, but it is actually only one character, with the \ indicating that what follows it is something special.
- '\0' is called null character.
- Note that '\0' and '0' are not same.
- ASCII value of '\0' is 0, whereas ASCII value of '0' is 48.
- The terminating null ('\0') is important, because it is the only way the functions that work with a string can know where the string ends.



- The string used above can also be initialized as,

  char name[ ] = "HAESLER" ;
- Note that, in this declaration ‘\0’ is not necessary.

Program to demonstrate printing of a string

```
main( )
{
   char name[ ] = "Klinsman" ;
   int i = 0 ;
   while ( i <= 7 )
   {
      printf ( "%c", name[i] ) ;
      i++ ;
   }
```

- Can we write the **while** loop without using the final value 7?

- Can we write the **while** loop without using the final value 7?
- We can; because we know that each character array always ends with a ‘\0’.

```
main( )
{
   char name[ ] = "Klinsman" ;
   int i = 0 ;
   while ( name[i] != `\0' )
   {
      printf ( "%c", name[i] ) ;
      i++ ;
    }
}
```

Input and Output of Character Array

```
main( )
{
   char name[25] ;
   printf ( "Enter your name " ) ;
   scanf ( "%s", name ) ;
    printf ( "Hello %s!", name ) ;
}
```

Output:
Enter your name Debashish
Hello Debashish!

While entering the string using **scanf( )** we must be cautious about two things:

- The length of the string should not exceed the dimension of the character array.
  - C compiler doesn't perform bounds checking on character arrays.
- **scanf( )** is not capable of receiving multi-word strings.
- Names such as 'Debashish Roy' would be unacceptable.
- The way to get around this limitation is by using the function **gets( )**.

gets() and puts()

```
main( )
{
   char name[25] ;
   printf ( "Enter your full name " ) ;
   gets ( name ) ;
   puts ( "Hello!" ) ;
   puts ( name ) ;
}
```

```
Enter your name
Debashish Roy
Hello!
Debashish Roy
```

gets() and puts()

- **puts( )** can display only one string at a time (hence the use of two **puts( )** in the program above).
- Also, on displaying a string, unlike **printf( )**, **puts( )** places the cursor on the next line.
- Though **gets( )** is capable of receiving only one string at a time, the plus point with **gets( )** is that it can receive a multi-word string.

Passing Array Elements to a Function using Call by Value

```
/* Demonstration of call by value */
main( )
{
   int i ;
   int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
   for ( i = 0 ; i <= 6 ; i++ )
         display ( marks[i] ) ;              // Function Call
 }

display ( int m )                            // Function Definition
{
   printf ( "%d ", m ) ;
}
```

Passing Array Elements to a Function using Call by Value

- Here, we are passing an individual array element at a time to the function **display( )** and getting it printed in the function **display( ).**
- Note that since at a time only one element is being passed, this element is collected in an ordinary integer variable **m**, in the function **display( )**.

Passing Array Elements to a Function –Using Call by Reference

```
/* Demonstration of call by reference */
main( )
{
  int i ;
  int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
  for ( i = 0 ; i <= 6 ; i++ )
      disp ( &marks[i] ) ;
}

disp ( int *n )
{
   printf ( "%d ", *n ) ;
 }
```

Passing Array Elements to a Function –Using Call by Reference

- Here, we are passing addresses of individual array elements to the function **display( )**.
- Hence, the variable in which this address is collected (**n**) is declared as a pointer variable.
- And since **n** contains the address of array element, to print out the array element we are using the 'value at address' operator **(*)**.

Passing an Entire Array to a Function

```
/* Demonstration of passing an entire array to a function */
main( )
{
  int num[ ] = { 24, 34, 12, 44, 56, 17 } ;
  dislpay ( &num[0], 6 ) ;
}

display ( int *j, int n )
 { int i ;
    for ( i = 0 ; i <= n - 1 ; i++ )
       {
           printf ( "\nelement = %d", *j ) ;
           j++ ; /* increment pointer to point to next element */
       }
}
```

Passing an Entire Array to a Function

- Here, the **display( )** function is used to print out the array elements.
- The address of the zeroth element is being passed to the **display( )** function.
- The **for** loop is same as the one used in the earlier program to access the array elements using pointers.
- It is also necessary to pass the total number of elements in the array, otherwise the **display( )** function would not know when to terminate the **for** loop.
- Note that the address of the zeroth element (many a times called the base address) can also be passed by just passing the name of the array.
- Thus, the following two function calls are same:

```
display ( &num[0], 6 ) ;
display ( num, 6 ) ;
```

Exercise

- Write a program for insertion in a sorted list
- Write a program for deletion in an unsorted list