Passing arrays to functions

- To pass a 1-dimensional array to a called function
- The name of the array **without any subscript** and the size of the array is passed as an argument to the function.
- **Syntax**:

function-name(array-name , size);

Example:

largest(value,4);

```
 float largest(float a[], int n);
void main()
{
        float  value[4]={2.5,6.5,66.8,98.2};
        printf("%f \n", largest(value,4));
}
 float largest(float a[], int n)
{
    int  i;
    float max;
    max=a[0];
    for(i=1;i<n ; i++)
    {
        if(max<a[i])
        max=a[i];
    }
        return(max);
}
```

- In C, the name of the array represents the address of its first element.
- Passing the array name, passing the address of the array to the called function. The array in the called function refers to the same array stored in the memory.
- Any changes in the array in the called function will be reflected in the original array

Rules to pass array to function

1. Function must be called by passing only the name of the array
2. In function definition, the formal parameter must be an array type, the array size must not be specified.
3. Function prototype must show the argument is an array.

Pointers and Character String

```
 void main()
{
 char name[] = “Klinsman”;
    char *ptr;
    ptr=name;
  while(*ptr!=‘\0’)
{
printf(“\n%c”*ptr);
 ptr++;
}
}
```

Character Pointer Arrray

- To store a string simply:

char str[8]=“Hello”;

- To ask C compiler to store it at some location

char *p=“Hello”,

Example

```
 void main()
{
    char str1[]=“hello”;
          char str2[10];
          char *s= “Good Morning”;
          char *q;

       str2=str1;
       q= s;
       return;
       }
```

Array of pointers

```
void main()
{
 int  *a[3];s
int i,j;
for(i=0;i<3;i++)
{
   a[i]=(int*)malloc(4*sizeof(int));
}
```

Array of pointers

- char name[3][25];
- int i;
- char *name[3] = {

  “New Zealand”,
  “Australia”,
  “India”
  };
- name is declared as an array of three pointers to characters.
- Each pointer pointing to a particular name

```
for(i=0;i<=2;i++)
  {
      printf("%s\n",name[i]);
  }
```

Difference between *p[3] and (*p)[3]

*p[3] declares p as an array of 3 pointers

(*p)[3] declares p as a pointer to an array of 3 elements