



# Arrays

---

## Session 7



# Objectives

---

- Explain array elements and indices
- Define an array
- Explain array handling in C
- Explain how an array is initialized
- Explain string / character arrays
- Explain two dimensional arrays
- Explain initialization of two dimensional arrays



# Array Elements & Indices

---

- Each member of an array is identified by unique index or subscript assigned to it
- The dimension of an array is determined by the number of indices needed to uniquely identify each element
- An index is a positive integer enclosed in [ ] placed immediately after the array name
- An index holds integer values starting with zero
- An array with 11 elements will look like -

**Player[0], player[1], player[2],.... Player[10]**



# Defining an Array-1

---

- An array has some particular characteristics and has to be defined with them
- These characteristics include –

*Storage Class*

*Data Types of the elements in the Array*

*Array Name* Which indicates the location of the first member of the array

*Array Size* a constant evaluating to a +ve value



# Defining an Array-2

---

An array is defined in the same way as a variable is defined. The only change is that the array name is followed by one or more expressions, enclosed within square brackets [], specifying the array dimension.

```
Storage_Class data_types array_name[size]  
int player[11];
```



# Norms with Arrays

---

- All elements of an array are of the same type
- Each element of an array can be used wherever a variable is allowed or required
- Each element of an array can be referenced using a variable or an integer expression
- Arrays can have their data types like **int, char, float or double**



# Array Handling in C-1

---

- An array is treated differently from a variable in C
- Two arrays, even if they are of the same type and size cannot be tested for equality
- It is not possible to assign one array directly to another
- Values cannot be assigned to an array on the whole, instead values are assigned to the elements of the array



# Array Handling in C-2

```
/* Input values are accepted from the user into the array ary[10]*/
#include <stdio.h>
void main()
{
    int ary[10];
    int i, total, high;
    for(i=0; i<10; i++)
    {
        printf("\n Enter value: %d : ", i+1);
        scanf("%d",&ary[i]);
    }
    /* Displays highest of the entered values */
    high = ary[0];
    for(i=1; i<10; i++)
    {
        if(ary[i] > high)
            high = ary[i];
    }
    printf("\nHighest value entered was %d", high);
    /*prints average of values entered for ary[10] */
    for(i=0,total=0; i<10; i++)
        total = total + ary[i];
    printf("\nThe average of the elements of ary is %d",total/i);
}
```





# Array Initialization

- Each element of an Automatic array needs to be initialized separately
- In the following example the array elements have been assigned values using the **for** loop

```
#include <stdio.h>

void main()
{
    char alpha[26];
    int i, j;
    for(i=65, j=0; i<91; i++, j++)
    {
        alpha[j] = i;
        printf("The character now assigned is %c \n", alpha[j]);
    }
    getchar();
}
```

- In case of extern and static arrays, the elements are automatically initialized to zero

# Strings/Character Arrays-1

- A string can be defined as a character type array, which is terminated by a null character
- Each character in a string occupies one byte and the last character of a string is "\0" (Backslash zero)
- Example

```
#include <stdio.h>
void main()
{
    char ary[5];
    int i;
    printf("\n Enter string : ");
    scanf("%s",ary);
    printf("\n The string is %s \n\n", ary);
    for (i=0; i<5; i++)
        printf("\t%d", ary[i]);
}
```

# Strings/Character Arrays-2

## Output -

If the entered string is appl, the output will be as shown below.

```
The string is appl
97  112  112  108  0
```

**The input for the above is of 4 characters and the 5<sup>th</sup> character is the null character**

If the entered string is apple, the output will be as shown below.

```
The string is apple
97  112  112  108  101
```

**The above output is for an input of 5 characters**



# String Functions

Wide range of string functions, which are found in the standard header file `<string.h>`

Name	Function
<code>strcpy(s1, s2)</code>	Copies <code>s2</code> into <code>s1</code>
<code>strcat(s1, s2)</code>	Concatenates <code>s2</code> onto the end of <code>s1</code>
<code>strlen(s1)</code>	Returns the length of <code>s1</code>
<code>strcmp(s1, s2)</code>	Returns 0 if <code>s1</code> and <code>s2</code> are the same; less than 0 if <code>s1 &lt; s2</code> ; greater than 0 if <code>s1 &gt; s2</code>
<code>strchr(s1, ch)</code>	Returns a pointer to the first occurrence of <code>ch</code> in <code>s1</code>
<code>strstr(s1, s2)</code>	Returns a pointer to the first occurrence of <code>s2</code> in <code>s1</code>



# Two-Dimensional Arrays

---

- The simplest and the most commonly used multi-dimensional array is the two - dimensional array
- A two-dimensional array can be thought of as an array of two single dimensional arrays
- A two-dimensional array looks like a railway time-table consisting of rows and columns
- A two-dimensional array is declared as -

**int temp[4][3];**



# Initialization of Multidimensional Arrays-1

```
int ary[3][4] =
```

```
{1,2,3,4,5,6,7,8,9,10,11,12};
```

The result of the above assignment will be as follows :

ary[0][0] = 1	ary[0][1] = 2	ary[0][2] = 3	ary[0][3] = 4
ary[1][0] = 5	ary[1][1] = 6	ary[1][2] = 7	ary[1][3] = 8
ary[2][0] = 9	ary[2][1] = 10	ary[2][2] = 11	ary[2][3] = 12



# Initialization of Multidimensional Arrays-2

---

```
int ary[3][4]=  
    {  
        {1,2,3},  
        {4,5,6},  
        {7,8,3}  
    };
```



# Initialization of Multidimensional Arrays-3

The result of the assignment will be as follows :

<code>ary[0][0] =1</code>	<code>ary[0][1]=2</code>	<code>ary[0][2]=3</code>	<code>ary[0][3]=0</code>
<code>ary[1][0]=4</code>	<code>ary[1][1]=5</code>	<code>ary[1][2]=6</code>	<code>ary[1][3]=0</code>
<code>ary[2][0]=7</code>	<code>ary[2][1]=8</code>	<code>ary[2][2]=3</code>	<code>ary[2][3]=0</code>

A two - dimensional string array is declared in the following manner :

```
char str_ary[25][80];
```





# Two-Dimensional Array-1

```
#include <stdio.h>
#include <string.h>
void main ()
{
    int i, n = 0;
    int item;
    char x[10][12];
    char temp[12];

    clrscr();
    printf("Enter each string on a separate line\n\n");
    printf("Type 'END' when over \n\n");

    /* read in the list of strings */
    do
    {
        printf("String %d : ", n+1);
        scanf("%s", x[n]);
    } while (strcmp(x[n++], "END"));
    /*reorder the list of strings */
```

**Example**

**contd...**



# Two-Dimensional Array-2

```
n = n - 1;
for(item=0; item<n-1; ++item)
{
    /* find lowest of remaining strings */
    for(i=item+1; i<n; ++i)
    {
        if(strcmp (x[item], x[i]) > 0)
        {
            /*interchange two strings */
            strcpy (temp, x[item]);
            strcpy (x[item], x[i]);
            strcpy (x[i], temp);
        }
    }
}

/* Display the arranged list of strings */
printf("Recorded list of strings : \n");
for(i = 0; i < n ; ++i)
{
    printf("\nString %d is %s", i+1, x[i]);
}
}
```

**Example**