PDS Lab

Section 16

Autumn-2018

Tutorial 4

Arrays

An array is a sequence of data item of homogeneous values (same type).

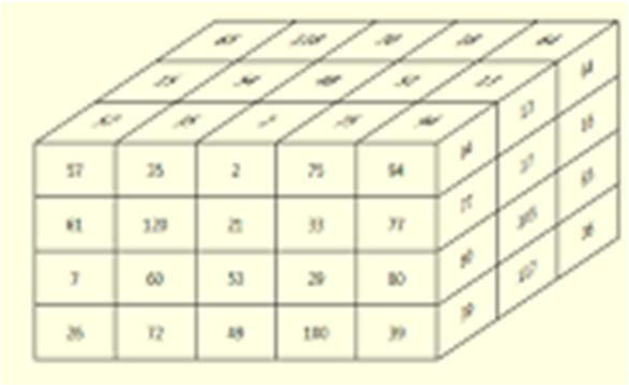
Arrays are of two types:

1. One dimensional arrays (1D array)
2. Multidimensional arrays (2D, #D, etc.)

1D array

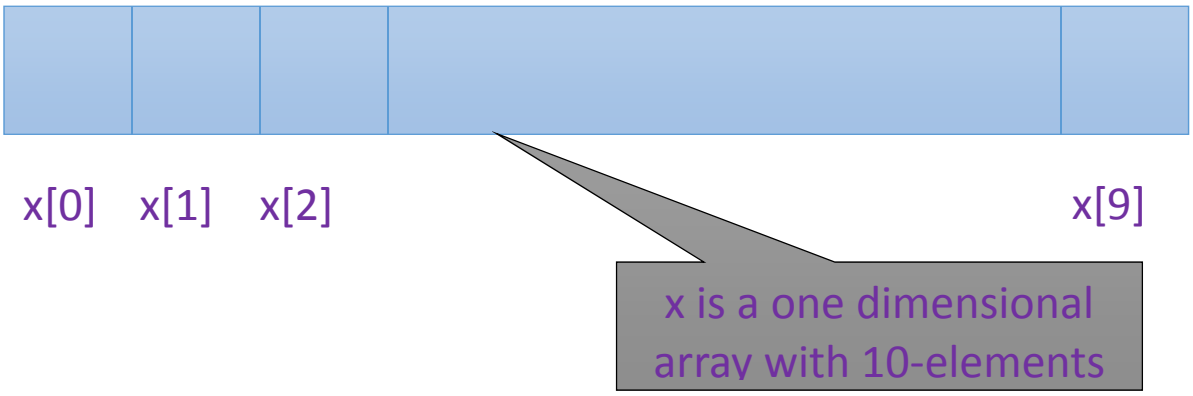
Routes	Mon 0	Tue 1	Wed 2	Thu 3	Fri 4
0	8	12	9	7	10
1	5	7	3	0	4
2	20	15	18	21	14
3	6	9	5	8	11

2D array



3D array

Example



Note:

- Homogenous data types
- All the data items constituting the group share the same name, that is, the name of the array.
- Individual elements are accessed by specifying the index.
- Range of indices varies between 0 and n-1 (both inclusive).

Declaration of Arrays

One Dimensional Array Declaration

Like variables, the arrays that are used in a program must be declared before they are used.

General syntax:

```
type   array-name [size];
```

`type` specifies the type of element that will be contained in the array (`int, float, char`, etc.)

`size` is an integer constant which indicates the maximum number of elements that can be stored inside the array.

Example

```
int   marks[50];
```

Here, `marks` is an array containing a maximum of 50 integers.

Be careful!
There is no element like `marks[50], marks[-1], marks[100],` etc.

Examples

```
int  x[10];

char  line[80];

float  points[150];

char  name[35];
```

Be careful!

The following kind of usage is illegal:

```
int   n;
int   marks[n];
```

Initialization of Arrays

One Dimensional Array Initialization

General form

```
type  array_name[size] = { list of values };
```

Example

```
int  marks[5] = {72, 83, 65, 80, 76};

char  name[4] = {‘A’, ‘m’, ‘i’, ‘t’};
```

Some special cases

1. If the number of values in the list is less than the number of elements, then the remaining elements are automatically set to zero.

Example

```
float  total[5] = {24.2, -12.5, 35.1};

 total[0]=24.2    total[1]=-12.5
 total[2]=35.1    total[3]=0.0
 total[4]=0.0
```

2. If the size declared is less than the elements in the initialization, then the excess elements will be ignored.

Example

```
float total[5] ={2.2,-1.5,3.5,0.1,0.2, 0.4};
```

The last element namely 0.4 will be ignored!

3. The size may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

Example

```
int   flag[] = {1, 1, 1, 0};

char  name[] = {‘A’, ‘m’, ‘i’, ‘t’};
```

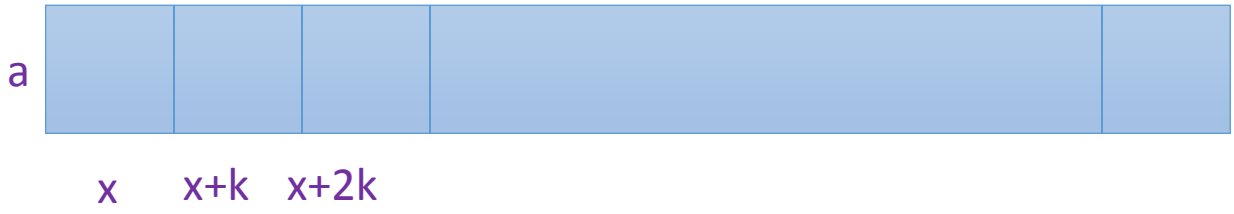
Initialization of array during runtime

Example

```
int marks[5];        //Declaration

for (int i=0; i < 5; ++i)
  scanf(“%d”,&marks[i]);  //Reading and storing
```

How an Array is Stored in Computer?

Starting from a given memory location, the successive array elements are allocated space in consecutive memory locations.



Here, `x` is the starting address of the array in memory.

The array index starts at zero. That is, the location of `a[0]` is at `x`

Let `k` is the number of bytes allocated per array element.

Element `a[i]` is allocated at the memory location having address `x + i*k`

Be careful!

In C, while accessing array elements, array bounds are not checked.

Example:

```
int   marks[5];
marks[8] = 75;
```

The above assignment would not report any error; however, execution will fail.

Rather, it may result in unpredictable program results.

Average Calculation

Example

Storing elements into an array and accessing them

```
int main()
 {
  int marks[50];
  int i=0, sum=0;

//Enter the marks obtained in a subject  and store them in the array

  for (i=0; i <50; ++i) {
    printf("Enter the mark for the %d-th ubject\n",i+1);
    scanf( "%d",&marks[i]);    // read an array element
  }


 // Calculate the average of the marks in the subject

 for (i=0; i <50; ++i)
     sum += marks[i];  //use an array element
     printf ( "Average Mark = %f \n ",sum/50);

 return 0;
}
```

2D Arrays

Example

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Student 1	75	82	90	65	76
Student 2	68	75	80	70	72
Student 3	88	74	85	76	80
Student 4	50	65	68	40	70

The table contains a total of 20 values, five in each line.

The table can be regarded as a matrix consisting of four rows and five columns.

C allows us to define such tables of items by using 2D arrays.

General form:

```
type   array_name [row_size][column_size];
```

Example

```
int  marks[4][5];

float  sales[12][25];

double  matrix[100][100];
```

Accessing Elements of 2D Array

Similar to that for 1D array, but use two indices.

Example

```
x[m][n] = 0;

c[i][k] += a[i][j] * b[j][k];
```

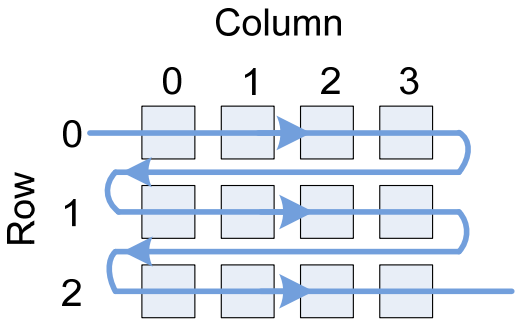
First indicates row, second indicates column.

Both the indices may be expressions which would evaluate to integer values.

```
x = sqrt (a[j*3][k+2]);
```

Strong 2D Arrays in Memory

Starting from a given memory location, the elements are stored row-wise in consecutive memory locations.



a[0]0] a[0][1] a[0]2] a[0][3] a[1][0] a[1][1] a[1][2] a[1][3] a[2][0] a[2][1] a[2][2] a[2][3]

Example

Suppose,

x: starting address of the array in memory
c: number of columns
k: number of bytes allocated per array element

Then
a[i][j] :: is allocated memory location at address

$$x + (i * c + j) * k$$

Reading the elements for 2D Array

Example

```
for(i=0; i < nrow; i++)
   for(j=0; j < ncol; j++)
       scanf("%f", &a[i][j]);
```

Reading a Matrix A_{mxn}

```
#include <stdio.h>
#define M 10;
#define N 15;

main ()
{
   int a[M][N];
   for (i=0; i<M; i++)
     for (j=0; j<N; j++)
         scanf("%d",&a[i][j]);
   }
```

Matrix Addition

```
#include <stdio.h>
#define m 10;
#define n 15;

main ()
{
   int a[m][n];
   int b[m[n];
    // Read the array a
    // Read the array b
   for (i=0; i<m; i++)
     for (j=0; j<m; j++)
         c[i][j] = a[i][j]+ b[i][j];
   }
```

Matrix Multiplication

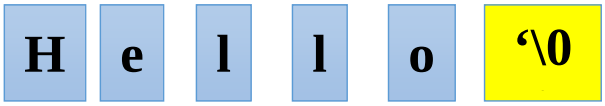
```
#include <stdio.h>
#define m 10;
#define p 5;
#define n 15;

main ()
{
   int a[m][p]; int b[p[n];
   int c[m][n];
    // Read the array a, b
   for (i=0; i<m; i++)
     for (j=0; j<n; j++)
        c[i][j] = 0;
        for (k=0; k<p; k++)
          c[i][j] = c[i][j]+ a[i][j]* b[i][j];
   }
```

Strings

- A string is an array of characters.
  - Individual characters are stored in memory in ASCII code.
  - A string is represented as a sequence of characters terminated by the null (‘\0’) character.

Example

“Hello” ➔



Example

- Declaration of strings

```
char   name[30];
char   city[ ];
char   *dob;
```

- A string may be initialized at the time of declaration.

```
char   city[15] = “Calcutta”;
char                                    city[15]
={‘C’,‘a’,‘l’,’c’,‘u’,‘t’,’t’,‘a’};
char   dob[] = “12-10-1975”;
```

Reading a line of text

- In many applications, we need to read in an entire line of text (including blank spaces).

- We can use the `getchar()` function for the purpose.

Example

```
char   line[81];
int  ch, c=0;
:
:
do
   {
       ch = getchar();
       line[c] = ch;
       c++;
   }
while  (ch != '\n');

line[c] = '\0';
```

Read characters until CR ('\n') is encountered

Make it a valid string

Writing a string on the screen

We can use `printf` with the "%s" format specification.

```
printf  ("\n %s", name);
```

Alternatively, we can use printf with the "%c" format and each character one after another.

```
i = 0;
while (name[i++] !='\0')
        printf("%c", name[i]);
```

String functions

There exists a set of C library functions for character string manipulation.

```
strcpy  ::  string copy
strlen  ::  string length
strcmp  ::  string comparison
strtcat ::  string concatenation
```

- It is required to add the line.

```
#include  <string.h>
```

strcpy()

- Works very much like a string assignment operator.

```
strcpy  (string1, string2);
```

  - Assigns the contents of `string2` to `string1`.

Example

```
strcpy  (city, “Calcutta”);
strcpy  (city, mycity);
```

- Warning:
  - Assignment operator do not work for strings.

```
city  =  “Calcutta”;
```
INVALID

strlen()

- Counts and returns the number of characters in a string.

```
len  =  strlen (string);  /* Returns an int */
```

- The null character (‘\0’) at the end is not counted.

Example

```
char  city[15];
int  n;
:
:
strcpy (city, “Calcutta”);
n = strlen (city);
```

n is assigned 8

strcmp()

Compares two character strings.

```
int  strcmp (string1, string2);
```

Compares the two strings and returns 0 if they are identical; non-zero otherwise.

Example

```
if   (strcmp (city, “Delhi”) = = 0)
    {  …… }

if  (strcmp (city1, city2) ! = 0)
    {  …… }
```

strcat()

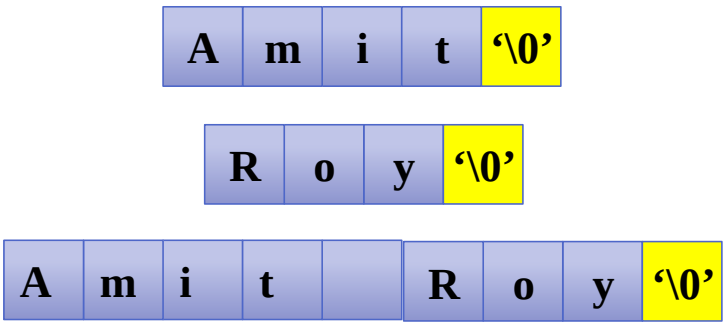
- Joins or concatenates two strings together.

  `strcat (string1, string2);`

  - `string2` is appended to the end of `string1`.
  - The null character at the end of string1 is removed, and `string2` is joined at that point.

Example

```
strcpy (name1, “Amit “);
strcpy (name2, “Roy“);
strcat(name1, name2);
```



Example

Read a text and count the number of uppercase letters in it.

```
#include  <stdio.h>
#include  <string.h>
main()
{
    char  line[81];
    int  i, n, count=0;
    scanf (“%[^\n]”, line);
    n = strlen (line);
    for  (i=0; i<n; i++)
       {
           if  (isupper (line[i])
                  count++;
       }
    printf (“\n The no. of uppercase ltr in the str %s
is %d”,line, count);
```

Example

Read a text and count the number of words and sentences in it.

```
#include  <stdio.h>
#include  <string.h>
#define MAX 1000;
main()
{
    char  myText[MAX];
    int  i, n, wCount=0; lCount = 0;
    while ((c=getchar()) != 0) myText[i++] = c;
    myText[i] = '\0';
    n = strlen (myText);
    for  (i=0; i<n; i++)
       {
           switch(myText[i]){
           case ' ': wCount++; break;
      case '\.' :lcount++; break;
       }
    printf ("\n The number of words is %d and sentences is
%d",wCount, lCount);
}
```

Tutorial Problems

Problem 1

Assume that array A and B are declared as follows:

```
int A[5][4];
int B[4];
```

Find the errors (if any), in the following program segments

a)
```
for (i=1; i<4; i++)
    scanf("%f", B[i]);
```

b)
```
for (i=1; i<=5; i++)
  for (j=1; i<=4; j++)
      A[i][j] = 0;
```

c)
```
for (i=4; i>=0; i--)
    scanf("%d", &B[i]);
```

Problem 2

What is the output printed by the following program?

```
#include<stdio.h>

void main(){

    int A[]={0,1,2,3,4,5,6,7};
    int n=8, step = 2, i, j, k, l, temp;

    for(i=0;i<n-step;i++){
        for(j=i;j<i+step;j++){
            temp = A[j];
            A[j] = A[j+1];
            A[j+1] = temp;
       }
            step = (step*2)- 1;
    }
    for(i=0; i<n; i++)printf("%d ",A[i]);
}
```

Problem 3

What is output of the following program?

```
main()
{
    int a[5] = { 5, 1, 15, 20, 25 };
    int i, j, k;

    i = ++a[1] ;
    j = a[1]++ ;
    k = a[i++] ;

    printf ("%d,%d,%d", i, j, k);
}
```

Problem 4

Write a program to merge two arrays into a single array.

Problem 5

Write a program which will remove all repeated elements except the first one.

Problem 6

Write a program to store a 2D array of integers into an 1D array of integers.

Problem 7

Write a program to calculate the value of a polynomial of degree *n* for a given value of *x*.

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

Hint: You should store the polynomial into an array.

Problem 8

Figure out the output of the below code. State relevant assumption if any during computation.
Assume necessary libraries have been included.

```
#include <stdio.h>
   int main()
  {
  char *s = "Harry Potter";
      s[0] = 'K';
      printf("%s", s);
      return 0;
  }
```

(a) Segmentation fault on GCC
(b) Karry
(c) Potter

(d) Karry Potter

Problem 9

Figure out the output of the below code. State relevant assumption if any during computation.
Assume necessary libraries have been included.

```
#include<stdio.h>
int main()
{
  char *str = "A for Apple"
              "B for Ball"
              "C for Cat";
   puts(str);
   return 0;
  }
```

(a) Compilation Error at Line 4
(b) A for Apple B for Ball C for Cat
(c) A for Apple " "B for Ball " "C for Cat
(d) A for Apple

Problem 10

Which of the following arrays among arr1, arr2, arr3 are terminated by a null character during storage in memory. State relevant assumption if any during computation.
Assume necessary libraries have been included.

```
#include<stdio.h>
   int main()
   {
    char arr1[] = "Five";
    char arr2[5] = "Five";
    char arr3[]= {'F', 'i', 'v', 'e'};
    return 0;
   }
```

(a) arr1 only
(b) arr1 & arr2
(c) arr2 & arr3
(d) arr3 only

Problem 11

Write a program to store 8 bits (0s, 1s) binary string. The program should print the decimal value for the string stored.

Problem 12

Read a 1D array containing n elements (n input by user) containing only 0s and 1s. Print the length of the longest run of 1s. For example, in the array 01011110011, the length is 4.

Problem 13

Read any 10 names of the cities in India and store then in an array of strings. Traverse the array to find the city, which has the maximum number of vowels in the name. In case, two or more cities qualify the same, then print all of them.

Problem 14

An anagram is a word or phrase formed by rearranging the letters of another word or phrase. For example, "carthorse" is an anagram of "orchestra". Write a program which reads two character strings of same length and prints whether they are anagrams of each other.

Important links:

http://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html