Chapter 6 ARRAYS

- Arrays are structured data types that hold multiple variables of the same data type, stored in a consecutive memory location in common heading.
- Array is a set of similar data (homogeneous data items) that shares the common name.
- The individual values in the array are called as elements.
- An array lets you declare and work with a collection of values of the same type.

UNIVERSITY Of KIGALI

6.1 0NE DIMENSIONAL ARRAYS

Structured collection of components, all of the same type. Structure given a single name. Individual elements accessed by index indicating relative position in collection. Type of elements stored in an array can be anything. Index of an array *must* be an integer

One dimensional arrays can be inline arrays representation or unicolumn arrays representation.

1. **Inline array** is a container that stores the data itself not pointers to data, this means there is no memory fragmentation, also for small data types(such as char, short, int, long).

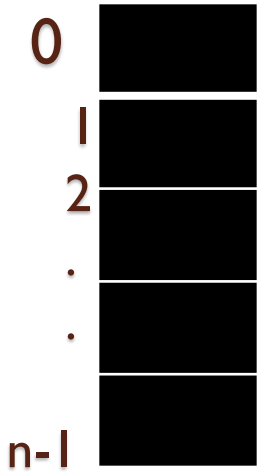
Example:

index 0 1 2 ……

n-1

Therefore it looks like this: int[ ] myArray = {0,1,2,3};

2. Unicolumn array



6.2. TWO DIMENSIONAL ARRAY

A two-dimensional array is really nothing more than an array of arrays, as a matrix. A matrix can be thought of as a grid of numbers, arranged in rows and columns, kind of like a bingo board. Therefore, it looks like this:

```
int[ ][ ] myArray = { {0,1,2,3}, {3,2,1,0}, {3,5,6,1}, {3,8,3,4} };
or int[ ][ ] myArray = { {0, 1, 2, 3},
                         {3, 2, 1, 0},
                         {3, 5, 6, 1},
                         {3, 8, 3, 4} };
```

12	34	56	787
	23		
		87	
			98

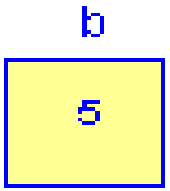
Example

- For example, you might want to create a collection of five integers.
- One way to do it would be to declare five integers directly: int a, b, c, d, e;
- This is okay, but what if you needed a thousand integers?
- An easier way is to declare an array of five integers:
- int a[5];
- The five separate integers inside this array are accessed by an **index**.
- All arrays start at index zero and go to n-1 in C.

UNIVERSITY Of KIGALI

A normal variable:

int b;



You place a value into b with the statement:

b=5;

An array variable:

int a[4];

You place a value into a with a statement like:

a[2]=5;

Declaring arrays

- Arrays are declared along with all other variables in the declaration section of the program.
- defining the type of array,
- name of the array,
- number of subscripts (whether is one or multi-dimensional)

UNIVERSITY Of KIGALI

Assigning initial values to arrays

- The initial values are enclosed in braces

UNIVERSITY Of KIGALI

20
8
12
15

Initialisation of array

0	a[0]=20
1	a[1]=8
2	a[2]=12
3	a[3]=15

Array declaration of one dimensional

syntax: data type array name [size];

Example: int a[4];

Accepting data for array:

input data into array always it requires to use loop in re

the size of the array, lets say n and the last index will be

syntax:

```
for(i=0; i<n; i++)
{
scanf("%d",&a[i]);
}
```

UNIVERSITY Of KIGALI

Accessing or display contents of arrays

It always support loops in order to perform this operation.

Syntax:

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

or

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

or

```
 i=0;
Do
{
printf("%d\n", a[i]);
i++;
}
while(i<n)
```

UNIVERSITY Of KIGALI

- **N.B:**
- Array whose elements are specified by one subscript are called *one dimensional array* or *single dimensional array.*
- If the maximum size of the array is 200 elements, then If you avail more than the declared size then the compiler will treat only the first n elements as significant.
- The subscript used to declare an array is sometimes called a ***dimension*** or **size** and the declaration for the array is often referred to as *dimensioning.*
- The dimension used to declare an array must always be a **positive integer** constant

UNIVERSITY Of KIGALI

EXERCISE TO DISPLAY POSITIVE INTEGER

```
#include<stdio.h>
main()
{
int a[7]={11,12,13,14,15,16,17};
int i;
printf("Contents of the array\n");
for(i=0;i<=6;i++)
printf("%d\t",a[i]);
}
```

UNIVERSITY Of KIGALI

2nd PROGRAM TO DISPLAY NUMBERS USING ARRAY

```
#include<stdio.h>
main()
{
int a[100];
int n,i;
printf("enter number of
integers \n");
scanf("%d",&n);
printf("enter those numbers
\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
  }
printf("the number
entered are: \n");
for(i=0;i<n;i++)
{
printf("values entered
=%d\n",a[i]);
}
}
```

UNIVERSITY Of KIGALI

3. PROGRAM TO PERFORM SUMMATION OF NUMBERS USING ARRAY

```
#include<stdio.h>
main()
{
int a[100]; int n,i; long int
sum=0;
printf("enter number of
integers \n");
scanf("%d",&n);
printf("enter those numbers
\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
sum=sum+a[i];
  }
printf("the number entered
are: \n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\n the Summation is:
%ld",sum);
}
```

UNIVERSITY Of KIGALI

4. PROGRAM TO PERFORM PRODUCT OF NUMBERS USING ARRAY

```
#include<stdio.h>
main()
{
int a[100]; int n,i; long int p=1;
printf("enter number of
integers \n");
scanf("%d",&n);
printf("enter those numbers
\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
p=p*a[i];
  }
printf("the number entered
are: \n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\n the product is:
%ld",p);
}
```

UNIVERSITY Of KIGALI

6.3 Multi Dimensioned Arrays

- Multi-dimensioned arrays have two or more index values, which specify the element in the array.
- multi[i][j]
- The first index value *i* specifies a row index, while *j* specifies a column index.
- You must remember that when we give values during one dimensional array declaration, we don't need to mention dimension.

  But that's not the case with 2D array; you must specify the second dimension even if you are giving values during the declaration. Let's understand this with the help of few examples.

```
/* Valid declaration*/
int abc[2][2] = {1, 2, 3 ,4 }
/* Valid declaration*/
int abc[ ][2] = {1, 2, 3 ,4 }
/* Invalid declaration – you must specify second dimension*/
int abc[ ][ ] = {1, 2, 3 ,4 }
/* Invalid because of the same reason  mentioned above*/
int abc[2][ ] = {1, 2, 3 ,4 }
```

6.3.1 INITIALIZATION OF 2D ARRAY

There are many ways to initialize two Dimensional arrays

```
int values[2][4] = {
                      {10, 11, 12, 13},
                      {14, 15, 16, 17}
                    };
```

or

```
int values[2][4] = { 10, 11, 12, 13, 14, 15, 16, 17};
```

UNIVERSITY Of KIGALI

- Now, consider the following array declaration:

  int values[3][4] ={1,2,3,4,5,6,7,8,9,10,11,12};

- The result of this initial assignment is as follows:

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

- This, can be initialized by forming groups of initials values enclosed within braces
- int values[3][4] ={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

- While initializing a two dimensional array, it is necessary to mention the second (column) dimension,
- the first dimension (row) is optional. Thus, the declarations given below are perfectly acceptable.
- int arr[3][4] ={12,34,23,45,56,45};
- int arr[ ][4] ={12,34,23,45,56,45};

UNIVERSITY Of KIGALI

2D array conceptual memory representation

Second subscript →

first subscript ↓

abc[0][0]	abc[0][1]	abc[0][2]	abc[0][3]
abc[1][0]	abc[1][1]	abc[1][2]	abc[1][3]
abc[2][0]	abc[2][1]	abc[2][2]	abc[2][3]
abc[3][0]	abc[3][1]	abc[3][2]	abc[3][3]
abc[4][0]	abc[4][1]	abc[4][2]	abc[4][3]

Here my array is A,B,C[5][4] which can be conceptually viewed a a matrix of 5 rows and 4 columns. Point to note here is that subscript starts with zero, which means ABC[0][0] would be the first value of the array.

1. Program that accept values in 2-Dimensional 3 by 3 array and displays the sum of all the elements.

```
#include<stdio.h>
main()
{
    int arr[3][3], i, j, sum=0;            /*Accepts input from the user and stores it in 2-D array*/
        for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("\nEnter the value for A[%d][%d]:",i,j);
            scanf("%d",&arr[i][j]);
        }
    }                                                   /*Calculate sum of elements in 2-D array*/
for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
                sum=sum+arr[i][j];
        }
    }                                                   /*Display the value of sum*/
    printf("\nThe sum of the elements of 2-D array is %d", sum);
Return 0;
}
```

UNIVERSITY Of KIGALI

6.3.2 INITIALIZATION OF MULTIDIMENSIONAL ARRAYS

In C, multidimensional arrays can be initialized in different number of ways.

```
int x[2][3]={{1,3,0}, {-1,5,9}};
```

OR

```
int x[][3]={{1,3,0}, {-1,5,9}};
```

OR

```
int x[2][3]={1,3,0,-1,5,9};
```

Suppose there is a multidimensional array arr[i][j][k][m]. Then this array can hold i*j*k*m numbers of data.

Similarly, the array of any dimension can be initialized in C programming.

UNIVERSITY Of KIGALI

2. Exercise to display multidimensional array in C

```
#include<stdio.h>
main()
{int a[4][4];
int i,j;
printf("Enter the 4*4 Matrix\n");
printf("_____________________________\n");
for(i=0;i<4;i++)
for(j=0;j<4;j++)
   scanf("%d",&a[i][j]);
   printf("The matrix entered is:\n");
         for(i=0;i<4;i++)
         for(j=0;j<4;j++)
         printf("a[%d][%d]=%d\n",i,j,a[i][j]);
 }
```

UNIVERSITY Of KIGALI

Example of Multidimensional Array In C

1. Write a C program to find sum of two matrix of order 2*2 using multidimensional arrays where, elements of matrix are entered by user.

```
#include <stdio.h>
int main(){
   float a[2][2], b[2][2], c[2][2];
   int i,j;
   printf("Enter the elements of 1st matrix\n");
   for(i=0;i<2;++i)
      for(j=0;j<2;++j){
      printf("Enter a%d%d: ",i+1,j+1);
      scanf("%f",&a[i][j]);
      }
   printf("Enter the elements of 2nd matrix\n");
   for(i=0;i<2;++i)
      for(j=0;j<2;++j){
      printf("Enter b%d%d: ",i+1,j+1);
      scanf("%f",&b[i][j]);
      }
   for(i=0;i<2;++i)
      for(j=0;j<2;++j){                    /* Writing the elements of multidimensional array using
loop. */
      c[i][j]=a[i][j]+b[i][j];             /* Sum of corresponding elements of two arrays. */
      }
   printf("\nSum Of Matrix:");
   for(i=0;i<2;++i)
      for(j=0;j<2;++j){
      printf("%.1f\t",c[i][j]);
         if(j==1)                          /* To display matrix sum in order. */
         printf("\n");
      }
return 0;
}
```