# LAB 8
# ARRAYS II

**NAME: EZRY BIN EDINOLFI**

**GROUP 1**

**Faculty of Electronic Engineering Technology**
**Universiti Malaysia Perlis**

## 1. OBJECTIVES:

**1.1** To be able to define an array, initialize an array and refer to individual elements of an array.
**1.2** To be able to pass arrays to functions.
**1.3** To be able to use arrays to store and process data in a program.

## 2. INTRODUCTION:

### 2.1 Passing Arrays to Functions

To pass an array argument to a function, specify the name of the array without any brackets. For example, if an array named **fnArray1** has been defined as

```
int aiArray1[15];
```

The function call: size of **aiArray1** is passed to **fnFunction1**

```
fnFunction1(aiArray1,15);
```

passes **aiArray1** and its size to function **fnFunction1**. Note the size of **aiArray1** is passed to the function, so the function can process the proper number of elements in the array.

In C, arrays are passed automatically to functions by reference – the called functions can modify the element values in the original arrays. The name of the array is actually the address of the first element of the array.

For a function to receive an array through a function call, the function header can be written as

```
void fnFunction1(int aiA[ ], int iSize)
```

indicating that **fnFunction1** expects to receive an array of integers in parameter **aiA** and the number of array elements in parameter **iSize**.

### 2.2 Passing Individual Array Elements to Functions

Although entire arrays are passed by reference in a function, individual array elements are passed by value. To pass an element of an array to a function, use the subscripted name of the array element as an argument in the function call.

The function call:

```
fnFunction2(aiArray1[2],15);
```

For a function to receive only an element of an array through a function call, the function header can be written as

```
void fnFunction2(int aiA, int iSize)
```

indicating that **fnFunction2** expects to receive an element of an array in parameter **a** exactly as simple variable is received, and also to receive the number of array elements in parameter **iSize**.

## 3. TASKS:

3.1

(a) Define a function fnMultiply that computes and returns the product of the type int element of its array input argument. The function should have a second input argument telling the number of array elements to use.

```
int fnMultiply ( int Arrai[i],int index)
{
   int i = index, j;
   int temp = Arrai

   for (j=0; j < i ; j++)
   {
      Arrai [i] *= temp [j];
   }
}
```

(b) Define a function fnAbsTable that takes an input array argument with type double values and display a table of the data and their absolute values like the table shown:

```
        x                 |x|
      38.4               38.4
    -101.7              101.7
      -2.1                2.1
        . . .
```

```
int fnAbsTable (double iPut [i],int index)
{
   int i = index;
   int temp [i], temp2 [i]
   for (j=0;j<i;j++)
   {
      if (iPut [j] < 0  )
         {
            temp [j] = iPut [j];
            temp2 [j] = temp [j]*-1 ;
         }
   }
   for (j=0;j<i;j++)
   {
      printf ("%d \t\t\t %d\n", temp[j],temp2[]j);
   }


}
```

(c) Write a function that negates the type integer values stored in an array. The first argument should be the array (an input/output parameter), and the second should be the number of elements to negate.

```
int minus (arr1[10],negate1[5])
{
   int i,j;
   for (i=0;i<10;i++)
   {
      for (j=0;j<10;j++)
      {
         if (negate1[i]=arr1[j])
            arr1[j]=0
      }
   }
}
```

(d) Write a function that takes two type int array input arguments and their effective size and produces a result array containing the absolute differences between corresponding elements. For example, for the three-element input arrays 5 -1 7 and 2 4 -2, the result would be an array containing 3 5 9.

```
int dif (int arrdif [10],int arr1[10],int arr2[10], n)
{
   int i;

   for (i=0;i<n;i++)
   {
      arrdif[i]=arr1[i]-arr2[i];
   }
}
```

(e) Write a function called fnReverse that takes an array named aiX as an input parameter and an array named aiY as an output parameter. A third function parameter is iN, the number of values in aiX. The function should copy the integers in aiX into aiY but in reverse order (i.e., aiY[0] →aiX[iN – 1], aiY[1] →aiX[iN – 2] . . . aiY[iN-1] →aiX[0])

```
int reverse (iX[10], iY[10],iN)
{
    int i, tmp;
    for( i = 0 ; i < iN / 2 ; i + + )
   {
      tmp = iX [ i ] ;
      iX [ i ] = iY [ iN - 1 - i ] ;
      arr [ iN - 1 - i ] = tmp ;
   }
}
}
```

3.2

(a) Write a C function that takes a single M x N integer matrix argument, and finds and returns the largest value in the matrix.

```
int largest (arr[10][10])
{
   int i,j;
   for ( i = 1; i < 5; ++i)
   {
     for (j = 1; j < 5; ++j)
     {
      if (arr[0][0] < arr[i][j])
      {
         arr[0][0] = arr[i][j];
      }
     }
   }
}
```

(b) Write a function that has two parameters--an M x N integer matrix, and an integer target value. The function should display the row and column subscripts of all occurrences of the target value.

```
int target (int arr[100][100],int arrTar[10],int m,int n)
{
   for (i=0;i<n;i++)
   {
      for (j=0;j<m;j++)
      {
         if (arr[i][j] == arrTar[j])
         {
            printf("The position of Target value is : [%d][%d]",i,j);
         }
      }
   }
}
```

(c). Write a function fill_mat that fills an N x N matrix of type double values with data from the keyboard.

```
int fill_mat (int arr[100][100],int n)
{
   int i,j;
   for (i=0;i<n;i++)
   {
      for (j=0;j<n;j++)
      {
        scanf("%d",arr[i][j]);
      }
   }
}
```

3.3 Define functions that:

(a) Create a new matrix by multiplying every element of its M x N matrix parameter by a scalar value.

```
int multiply (int arrSum [100], int arr [100] , int row; int column)
{
   int i,j;

   for(i=0;i<row;i++)
   {
     for (j=0;j<column;j++)
     {
        arrSum [i] = arr[i] * arr[j]
     }
   }
}
```

(b) Compute the product of an MxN matrix with an N-dimensional vector. The result is an M- dimensional vector.

```
int product (sum[100],int arr1 [100][100],int arr[100],int M,int N)
{
   for (i=0;i<M;i++)
   {
      for (j=0;j<N;j++)
      {
         sum[i]= arr1[i][j] * arr[j];
      }
   }
}
```

(c) Create a new M x P matrix C by multiplying every element of its M x N matrix A parameter by a N x P matrix B.

```
int SumMulti (int matrixSum[10][10],int mat1[10][10],int mat2[10][10],int M,int N,int P)
{
   int i,l,j,k;
   for (i=0;i<M;i++)
   {
      for (l=0;l<N;l++)
      {
         for (j=0;j<P;j++)
         {
            for (k=0;k<N;k++)
            {
               matrixSum [l][j]+= mat1[i][j] * mat2[i][k];
            }
         }
      }

   }
}
```

3.3 Write a program to store an input list of five numbers in an array named **`list`** and display the largest element in the array using a function named **`get_max`**. The function **`get_max`** will use the array and its size as input parameters and then returns the largest element in the array.
Sample Output:

```
Enter five numbers : 11 99 10 56 7

Element in array list[0] = 11
Element in array list[1] = 99
Element in array list[2] = 10
Element in array list[3] = 56
Element in array list[4] = 7

Largest element in array list : 99
```

```
#include <stdio.h>

int list (int iPut[5]);
void get_max (int arr[5]);

int main ()
{
   int arr[5];
   list (arr);
   get_max (arr);

}

int list (int iPut[5])
{
   int i;
   printf("Input five numbers:");

   for (i=0;i<5;i++)
   {
     scanf("%d",&iPut[i]);
   }


   for (i=0;i<5;i++)
   {
      printf("Element of Array list [%d] = %d\n",i,iPut[i]);
   }

}

void get_max (int arr[5])
{
   int i;
   for (int i = 1; i < 5; ++i)
   {
      if (arr[0] < arr[i])
      {
         arr[0] = arr[i];
      }
   }
   printf("Largest element is : %d", arr[0]);
}
```

3.4. Write a program that can calculate the sum of three equality M x N sized matrices, [MatrixA] + [MatrixB] + [MatrixC].

Program Specifications: Create functions 1) to read the input data. 2) to perform the calculation and 3) to display the result.

```
#include <stdio.h>

int get_Data (int arr[100][100], int,int);
int calc (int arr_tot[100][100],int arr1[100][100],int arr2[100][100],int arr3[100][100],int,int);
void out_display (int arr[100][100], int ,int);

int main ()
{
    int row,column, arr1 [100][100],arr2 [100][100],arr3 [100][100],arr_tot [100][100];

    printf("Please determine the size of Matrix (row & column):");
    scanf("%d%d",&row,&column);

    printf("\nEnter Element Array 1\n");
    get_Data (arr1,row,column);

    printf("\nEnter Element Array 2\n");
    get_Data (arr2,row,column);

    printf("\nEnter Element Array 3\n");
    get_Data (arr3,row,column);

    calc (arr_tot,arr1,arr2,arr3,row,column);

    out_display (arr_tot,row,column);

}

int get_Data (int arr[100][100], int row,int column)
{
    int i,j,x,y;

    x = row;
    y = column;



    printf("Please input data for the array\n");
    for (i=0;i<x;i++)
    {
        for (j=0;j<y;j++)
        {
            printf ("Element in array [%d][%d]:", i,j);
            scanf("%d",&arr[i][j]);
        }
    }
}
```

```
int calc (int arr_tot[100][100],int arr1[100][100],int arr2[100][100],int arr3[100][100],int
row,int column)
{
   int i,j,x,y;

   x = row;
   y = column;

   for (i=0;i<x;i++)
   {
      for (j=0;j<y;j++)
      {
        arr_tot[i][j]= arr1[i][j] + arr2[i][j] + arr3[i][j];
      }
   }

}

void out_display (int arr[100][100], int row,int column)
{
   int i,j,x,y;

   x = row;
   y = column;

   printf ("\n\nThe Sum of Matrix is:\n");
   for (i=0;i<x;i++)
   {
      for (j=0;j<y;j++)
      {
        printf ("%d  ",arr[i][j]);

        if (j == y -1)
        {
           printf ("\n");
        }
      }
   }
}
```