Name : Chovatiya Avikumar
SRN : PES1PG25CA044
Date: 08/10/2025
Course: UQ25CA652A, Data Structure

Unit – 1 : Experience learning

## 1. LEVELT – 1 Size of data type

- **Objective** : Write a program to determine and print the sizes of different data types in C.
- **Instructions:**
  - Use the sizeof operator to find the sizes of the following data types: int, float, double, char, and long.
  - Print the sizes in bytes.
- **Code:**

```
#include<stdio.h>

int main(){

printf("Size of int : %d bytes\n",sizeof(int));
printf("Size of float : %d bytes\n",sizeof(float));
printf("Size of double : %d bytes\n",sizeof(double));
printf("Size of char : %d bytes\n",sizeof(char));
printf("Size of long : %d bytes\n",sizeof(long));
return 0;
}
```

- **Output:**

```
Size of int : 4 bytes
Size of float : 4 bytes
Size of double : 8 bytes
Size of char : 1 bytes
Size of long : 4 bytes
```

## 2. LEVEL – 1 Pointer and arrays

- **Objective :** write a program to demonstrate pointer arithmetic with a 1-dimensional array.
- **Instructions.**
  - Declare a 1-dimensional array of integers and initialize it with 5 values.
  - Use a pointer to traverse the array and print each element using pointer arithmetic.
- **Code:**

```
#include<stdio.h>

int main(){
	int a[5]={10,20,30,40,50};
	int *ptr = a;
	for(int i=0;i<5;i++){
	   printf("%d,",*(ptr+i));
	}
return 0;
}
```

- **Output:**

10,20,30,40,50,

## 3. LEVEL – 1 2-Dimensional Array

- **Objective :** create a program that manipulate a 2-dimensional array.
- **Instruction :**
  - Declare and initialize a 2-dimensional array.
  - Write a function to calculate tha sum of elements of the matrix and return the result.
  - Print the sum.
- **Code:**

```
#include<stdio.h>

int main(){

    int a[3][3]={{10,20,30},{40,50,60},{70,80,90}};
    int result = 0;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            result += a[i][j];
        }
    }
    printf("sum of all elements : %d",result);
    return 0;
}
```

- **Output:**

```
sum of all elements : 450
```

4. **LEVEL – 1 : Calculating address in arrays**

- **Objective :** create a program that calculates the address of a specific element in both 1-dimensional and 2-dimensional array using their respective formulas.
- **Instruction:**
  - 1-dimensional array:
    - Declare and initialize a 1-dimensional array with 5 elements.
    - Write a function to calculate the address of an element using the formula:

      $$\text{Address} = \text{Base Address} + (i \times \text{Size of element})$$

      where $i$ is the index of the element
    - Print the address of a specified element.
  - 2-dimensional array:
    - Declare and initialize a 2-dimensional array with 3 rows and 3 columns.
    - Write a function to calculate the address of an element using the formula for row-major order:

      $$\text{Address} = \text{Base Address} + [(i \times \text{Total Columns}) + j] \times \text{Size of element}$$

      where $i$ and $j$ represent the row and column indices of the element.
    - Print the address of a specified element.
- **Code**

```
#include<stdio.h>

int main(){

    int arr1d[5]={1,2,3,4,5};
    int arr2d[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    int baseaddress1d = 2000;
    int baseaddress2d = 3000;
    int sizeof1d = sizeof(arr1d[0]);
    int sizeof2d = sizeof(arr2d[0][0]);
    int row = 1,col = 2;
    int index = 2;
```

```
    int address1d = baseaddress1d + (index * sizeof1d);
    int address2d = baseaddress2d + ((row * col+1) + col) * sizeof2d;

    printf("address of element at index %d in 1D array:
%d\n",index,address1d);
    printf("address of element at (%d, %d) in 2D array:
%d\n",row,col,address2d);

    return 0;
}
```

- **Output:**

    address of element at index 2 in 1D array: 2008
    address of element at (1, 2) in 2D array: 3020

## 5. LEVEL – 1 Array as Parameters:

- **Objective:** write a program that uses array as function parameters.
- **Instructions:**
  - Create a function that takes an array of integers and its size as parameters.
  - The function should calculate and return the average of the elements in the array.
  - In the main function, declare an array, initialize it, and call the average function.
- **Code:**

```
#include<stdio.h>

float calavg(int arr[],int size){
    int total = 0;
    for(int i=0;i<size;i++){
        total += arr[i];
    }
    float avg = total / size;
    return avg;
}

int main(){
    int arr[5] = {10,20,30,40,50};
    float avg = calavg(arr,5);
    printf("average of element %.1f\n",avg);
    return 0;
}
```

- **Output :**

```
average of element 30.0
```

## 6. LEVEL – 2 : Structures and Array of structures

- **Objective :** create a program that demonstrates the use of structures and passing them as parameters.
- **Instructions:**
  - Define a structure called student with fields for name, age, and marks.
  - Create an array of student structures and initialize it with sample data.
  - Write a function that takes the array of student structures and the size of the array as parameters to calculate the average marks.
- **Code**

```
#include<stdio.h>

struct student
{
   char name[50];
   int age;
   float mark[4];
};

float calavg(struct student s[],int size){

   float total = 0;

   for(int i = 0;i<size;i++){
      total = 0;
      for(int j = 0;j<4;j++){
         total += s[i].mark[j] / 4;
      }
      printf("average mark of student %d is %0.1f\n",i+1,total);
   }

   return 0;
```

```
}

int main(){

    struct student s[3] = {
        {"avi",20,{90,80,60,40}},
        {"parshant",20,{90,40,70,40}},
        {"darshit",19,{90,50,70,40}}
    };

    int size = 3;
    calavg(s,size);

    return 0;
}
```

- **OutPut:**

```
average mark of student 1 is 67.5
average mark of student 2 is 60.0
average mark of student 3 is 62.5
```

Sincerely yours,
Chovatiya Avikumar,
PES1PG25CA044