DSA Lab Experiment: 02

Lab Assignment 1: Understanding Union vs Structure

Problem Statement: Write a program to define a union and structure to store employee information (name, employee ID, and salary). Demonstrate the difference in memory usage and behavior between a union and structure when storing the same set of data.

Assignment Tasks:

- **Define a union and a struct for employee information.**
- **Initialize and display values stored in both the union and struct.**
- **Calculate and display the memory size occupied by each using sizeof()**

DSA Lab Experiment: 02

Solution :

main.c

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

// Define a structure for employee information
struct EmployeeStruct {
    char name[50];
    int employeeID;
    float salary;
};

// Define a union for employee information
union EmployeeUnion {
    char name[50];
    int employeeID;
    float salary;
};

int main() {
    // Initialize structure and union variables
    struct EmployeeStruct empStruct;
    union EmployeeUnion empUnion;

    // Assign values to the structure
    strcpy(empStruct.name, "Karan");
    empStruct.employeeID = 24687;
    empStruct.salary = 85000.0;

    // Assign values to the union
    strcpy(empUnion.name, "Karan");
    empUnion.employeeID = 24687;
    empUnion.salary = 85000.0;

    // Display the values in the structure
    printf("Structure:\n");
    printf("Name: %s\n", empStruct.name);
    printf("Employee ID: %d\n", empStruct.employeeID);
    printf("Salary: %.2f\n", empStruct.salary);

    // Display the values in the union
    printf("\nUnion:\n");
    printf("Name: %s\n", empUnion.name);
    printf("Employee ID: %d\n", empUnion.employeeID);
    printf("Salary: %.2f\n", empUnion.salary);

    // Display the size of structure and union
    printf("\nMemory Usage:\n");
    printf("Size of structure: %lu bytes\n", sizeof(empStruct));
    printf("Size of union: %lu bytes\n", sizeof(empUnion));

    return 0;
}
```

Output :

```
Structure:
Name: Karan
Employee ID: 24687
Salary: 85000.00

Union:
Name:
Employee ID: 1202062336
Salary: 85000.00

Memory Usage:
Size of structure: 60 bytes
Size of union: 52 bytes


...Program finished with exit code 0
Press ENTER to exit console.
```

DSA Lab Experiment: 02

Lab Assignment 2: Dynamic Memory Allocation with malloc() and free()

Problem Statement: Write a program to dynamically allocate memory for an array of integers. Perform the following operations:

1. Input the number of elements (n).

2. Allocate memory dynamically using malloc().

3. Input n elements into the array.

4. Find the sum and average of the elements.

5. Release the memory using free().

Assignment Tasks:

- **Use malloc() for dynamic memory allocation.**
- **Input values into the dynamically allocated array.**
- **Calculate sum and average.**
- **Use free() to release the allocated memory**

DSA Lab Experiment: 02

Soulution :

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n;
    int *arr;
    int sum = 0;
    float average;

    // Step 1: Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Step 2: Allocate memory dynamically using malloc()
    arr = (int *)malloc(n * sizeof(int));

    // Check if memory allocation was successful
    if (arr == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    // Step 3: Input n elements into the array
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }

    // Step 4: Calculate the sum and average of the elements
    average = (float)sum / n;

    // Display the sum and average
    printf("Sum of elements: %d\n", sum);
    printf("Average of elements: %.2f\n", average);

    // Step 5: Release the memory using free()
    free(arr);

    return 0;
}
```

Output:

```
Enter the number of elements: 5
Enter 5 elements:
1
2
3
4
5
Sum of elements: 15
Average of elements: 3.00


...Program finished with exit code 0
Press ENTER to exit console.
```