

Student Information

- Name:- Aditya Kumar
- Sap Id :- 590015145
- Branch :- M.C.A
- Batch :- B1
- Instructor :-Dr. Sourbh Kumar

Lab Assignment 1: Understanding Union vs Structure

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

```
union Employee {
    char name[50];
    int id;
    float salary;
};
```

```
struct EmployeeInfo {
    char name[50];
    int id;
    float salary;
};
```

```
int main() {
    union Employee emp1;
    struct EmployeeInfo emp2;

    // Assign values to union members
    strcpy(emp1.name, "Alice");
    emp1.id = 1234;
    emp1.salary = 50000.0;

    // Assign values to struct members
    strcpy(emp2.name, "Bob");
    emp2.id = 5678;
    emp2.salary = 60000.0;

    printf("Union Employee Details:\n");
    printf("Name: %s\n", emp1.name);
    printf("ID: %d\n", emp1.id);
    printf("Salary: %.2f\n", emp1.salary);
```

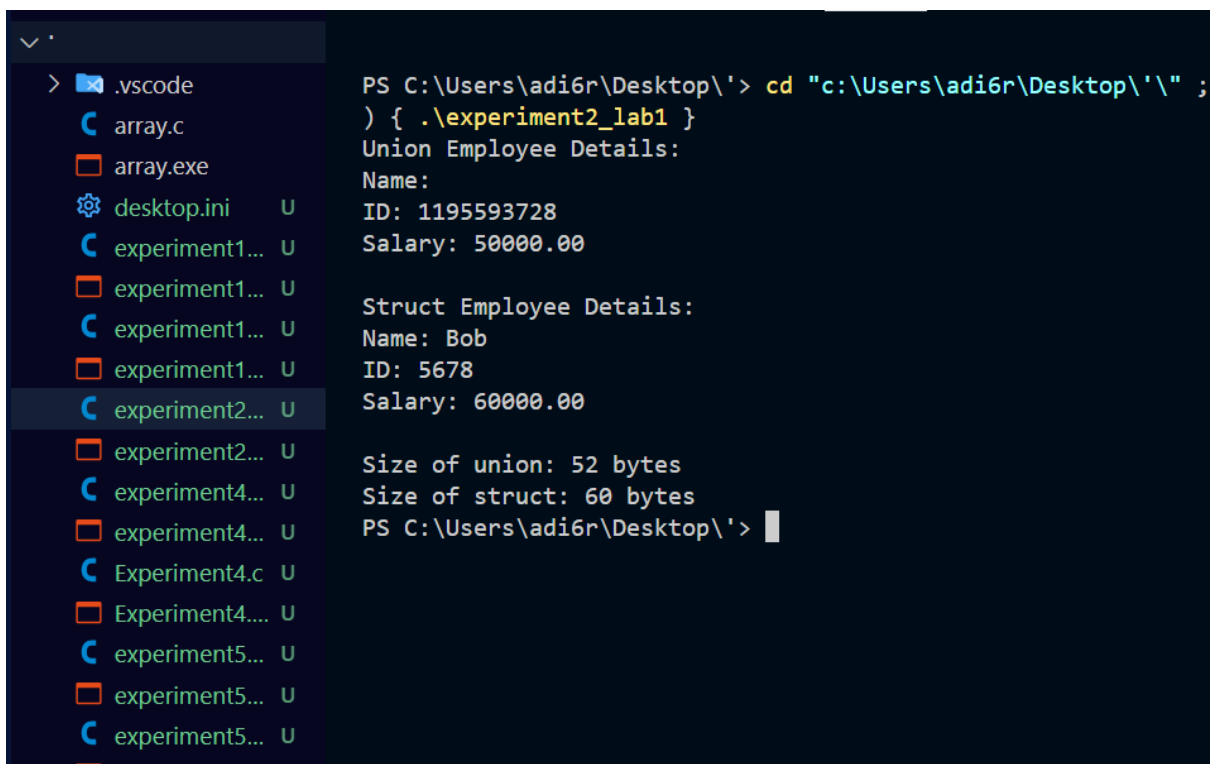
```

printf("\nStruct Employee Details:\n");
printf("Name: %s\n", emp2.name);
printf("ID: %d\n", emp2.id);
printf("Salary: %.2f\n", emp2.salary);

printf("\nSize of union: %zu bytes\n", sizeof(emp1));
printf("Size of struct: %zu bytes\n", sizeof(emp2));

return 0;
}

```



```

PS C:\Users\adi6r\Desktop\> cd "c:\Users\adi6r\Desktop\" ;
) { .\experiment2_lab1 }
Union Employee Details:
Name:
ID: 1195593728
Salary: 50000.00

Struct Employee Details:
Name: Bob
ID: 5678
Salary: 60000.00

Size of union: 52 bytes
Size of struct: 60 bytes
PS C:\Users\adi6r\Desktop\>

```

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

```

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

int main()
{
    int n;

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

```

```

// Dynamically allocate memory for an array of n integers
int *arr = (int*)malloc(n * sizeof(int));

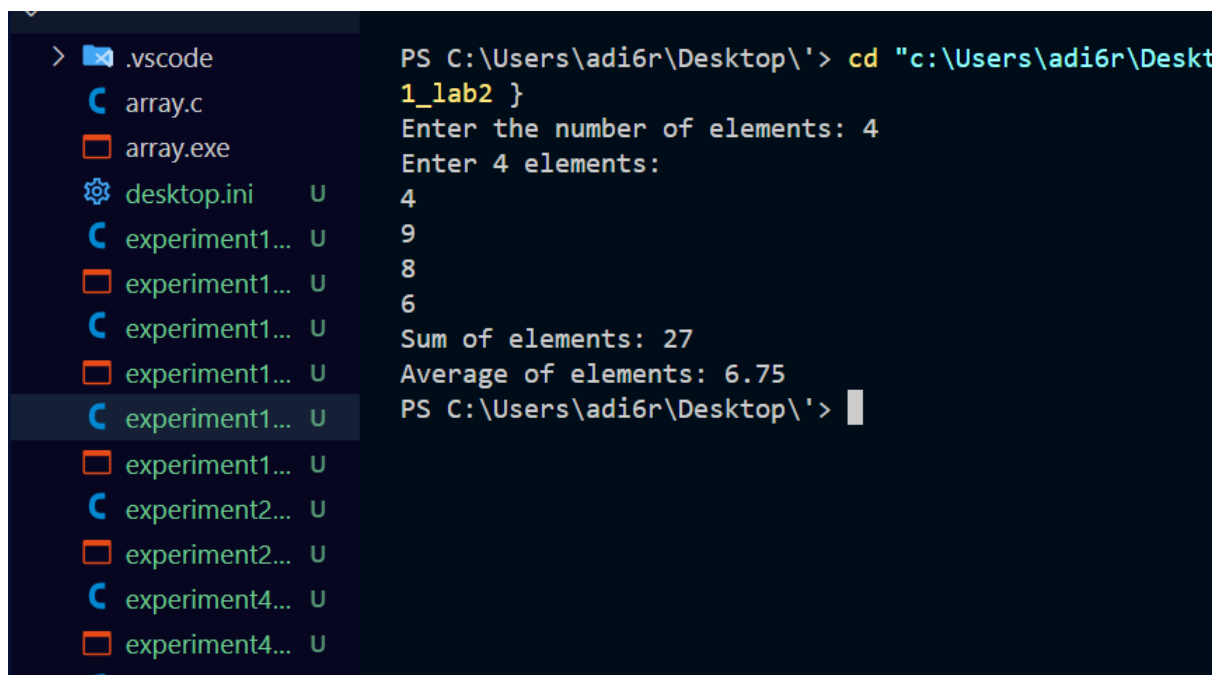
if (arr == NULL)
{
    printf("Memory allocation failed!\n");
    return 1;
}

printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
}
int sum = 0;
for (int i = 0; i < n; i++)
{
    sum += arr[i];
}
float average = (float)sum / n;
printf("Sum of elements: %d\n", sum);
printf("Average of elements: %.2f\n", average);

// Free the allocated memory
free(arr);

return 0;
}

```



The screenshot shows a VS Code editor with a file explorer on the left and a terminal on the right. The file explorer shows a project named '.vscode' with files 'array.c', 'array.exe', 'desktop.ini', and several 'experiment1...' and 'experiment2...' files. The terminal shows the execution of the program. The user has run 'cd "c:\Users\adi6r\Desktop\1_lab2"' and then executed the program. The program prompts for the number of elements (4) and then for 4 elements (4, 9, 8, 6). It then outputs the sum of elements (27) and the average of elements (6.75).

```

PS C:\Users\adi6r\Desktop\1_lab2> cd "c:\Users\adi6r\Desktop\1_lab2"
Enter the number of elements: 4
Enter 4 elements:
4
9
8
6
Sum of elements: 27
Average of elements: 6.75
PS C:\Users\adi6r\Desktop\1_lab2>

```