Course Name:	**Structured Programming Methodology**	**Semester:**	**I**
Date of Performance:	**11/ 11/ 2025**	**DIV/ Batch No:**	**A1**
Student Name:	**Arav Arun**	**Roll No:**	**16010125013**

Experiment No: 7
Title: Structures and unions

Aim and Objective of the Experiment:

To understand the concept of structure and union in C++ and to learn how to define, initialize, and access their members.

COs to be achieved:

CO4: Design modular programs using functions and the use of structure.

Theory:

Introduction to Structures

A structure in C++ is a user-defined data type that groups related variables of different data types under a single name.

Syntax:

```
struct StructureName {
   data_type member1;
   data_type member2;
   ...
};
```

Members are accessed using the dot (.) operator:

```
StructureName variable;
variable.member1 = value;
```

Example:

```
struct Student {
   int roll;
   char name[30];
```

```
    float marks;
};

int main() {
    Student s1 = {101, "Rahul", 88.5};
    cout << "Roll: " << s1.roll << "\n";
    cout << "Name: " << s1.name << "\n";
    cout << "Marks: " << s1.marks << "\n";
}
```

Introduction to Unions

A union is also a user-defined data type that allows storing different data types in the same memory location.

Only one member of a union can hold a value at a time because all members share the same memory.

Syntax:

```
union UnionName {
    data_type member1;
    data_type member2;
    ...
};
```

Example:

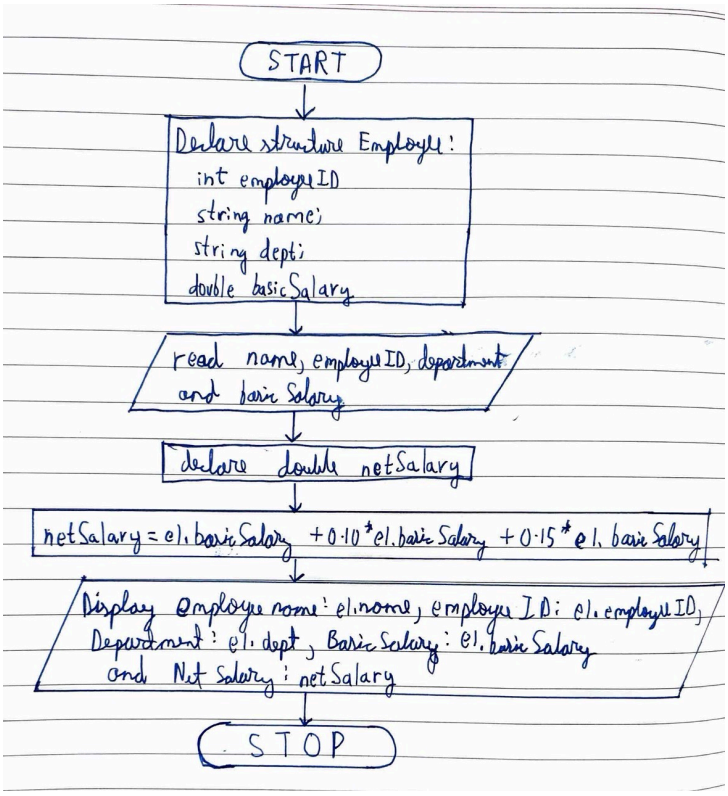
```
union Number {
    int x;
    float y;
};

int main() {
    Number n;
    n.x = 10;
    cout << "x = " << n.x << endl;

    n.y = 3.14;
    cout << "y = " << n.y << endl;
     cout << "x = " << n.x << endl; // Value of x changes because memory is shared
}
```

Problem Statements:

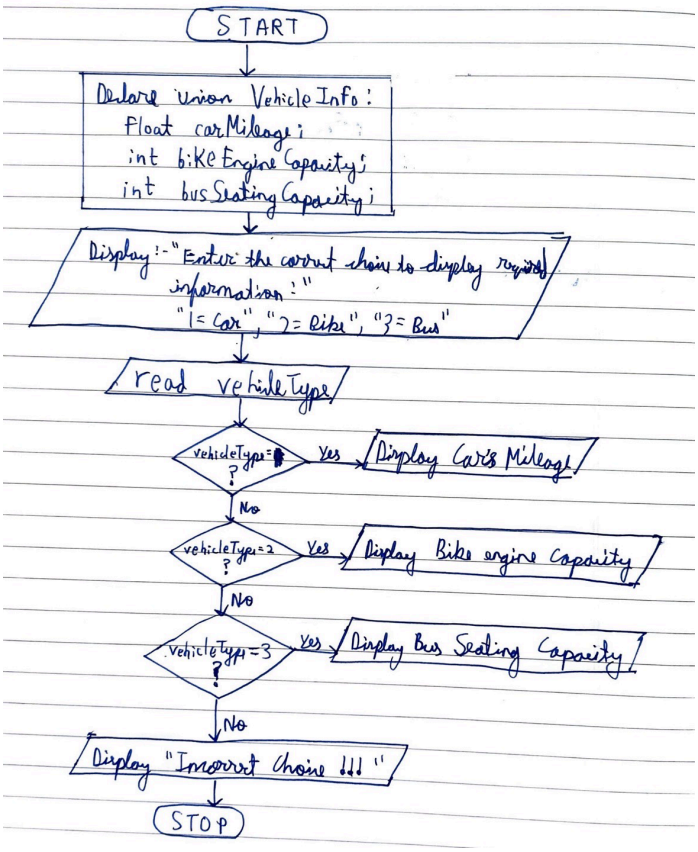
1. Write a C++ program using a struct Employee to store details such as employee ID, name, department, and basic salary. Compute and display the net salary using DA (10%) and HRA (15%). Hint: Use member variables and calculate net_salary = basic + 0.10*basic + 0.15*basic.
2. Create a union VehicleInfo that can store either a car's mileage (float), a bike's engine capacity (int), or a bus's seating capacity (int). Use an additional variable type (1 = Car, 2 = Bike, 3 = Bus) to determine which union member to use and display the correct information.

Algorithm / Flowchart:

Flowchart for program 1 :



Flowchart for program 2 :



Code :

Program 1 :

```
#include <iostream>
using namespace std;

struct Employee
{
   int employeeID;
   string name;
   string dept;
   double basicSalary;
};

int main()
{
   Employee e1;
   cout<<"Enter your name : "<<endl;
```

```
    cin>>e1.name;
    cout<<"Enter  your Employee ID : "<<endl;
    cin>>e1.employeeID;
    cout<<"Enter your department : "<<endl;
    cin>>e1.dept;
    cout<<"Enter basic salary : "<<endl;
    cin>>e1.basicSalary;

    double netSalary;
    netSalary = e1.basicSalary + 0.10*e1.basicSalary + 0.15*e1.basicSalary;

    cout<<"Name of employee : "<<e1.name<<endl;
    cout<<"Employee ID : "<<e1.employeeID<<endl;
    cout<<"Department :  "<<e1.dept<<endl;
    cout<<"Basic Salary :  "<<e1.basicSalary<<endl;
    cout<<"Net Salary of Employee : "<<netSalary<<endl;

    return 0;
}
```

Program 2 :

```
#include <iostream>
using namespace std;

union VehicleInfo
{
    float carMileage;
    int bikeEngineCapacity;
    int busSeatingCapacity;
};

int main()
{

    VehicleInfo v;
    int vehicleType;
    cout<<"Enter the correct choice to display required information : "<<endl;
    cout<<"1 = Car"<<endl;
    cout<<"2 = Bike"<<endl;
    cout<<"3 = Bus"<<endl;
    cin>>vehicleType;
```

```
    switch(vehicleType)
    {
        case 1:
            cout<<"Enter car's mileage : "<<endl;
            cin>>v.carMileage;
            cout<<"Car's mileage : "<<v.carMileage<<endl;
            break;
        case 2:
            cout<<"Enter bike engine capacity : "<<endl;
            cin>>v.bikeEngineCapacity;
            cout<<"Bike engine capacity : "<<v.bikeEngineCapacity<<endl;
            break;
        case 3:
            cout<<"Enter bus seating capacity : "<<endl;
            cin>>v.busSeatingCapacity;
            cout<<"Bus seating capacity : "<<v.busSeatingCapacity<<endl;
            break;
        default:
            cout<<"Incorrect Choice !!!"<<endl;
            break;
    }
    return 0;
}
```

Output:

Output for Program 1 :

```
Enter your name :
Arav
Enter  your Employee ID :
123456
Enter your department :
comp
Enter basic salary :
5000

Name of employee : Arav
Employee ID : 123456
Department :  comp
Basic Salary :  5000
Net Salary of Employee : 6250

Process returned 0 (0x0)   execution time : 9.507 s
Press any key to continue.
```

Output for Program 2 :

```
Enter the correct choice to display required information :
1 = Car
2 = Bike
3 = Bus
1
Enter car's mileage :
50
Car's mileage : 50

Process returned 0 (0x0)   execution time : 3.497 s
Press any key to continue.
```

```
Enter the correct choice to display required information :
1 = Car
2 = Bike
3 = Bus
2
Enter bike engine capacity :
25
Bike engine capacity : 25

Process returned 0 (0x0)   execution time : 6.437 s
Press any key to continue.
```

```
Enter the correct choice to display required information :
1 = Car
2 = Bike
3 = Bus
3
Enter bus seating capacity :
50
Bus seating capacity : 50

Process returned 0 (0x0)   execution time : 2.585 s
Press any key to continue.
```

Post Lab Subjective/Objective type Questions:

1. What is the difference between a structure and a union in C?

Sol:

Point of Difference	Structure	Union
1.Keyword	The keyword 'struct' is used to define a structure.	The keyword 'union' is used to define a union.
2.Size	When a variable is associated with a structure, memory is allocated for each member.The total size is greater than or equal to the sum of sizes of its members.	When a variable is associated with a union, memory is allocated equal to the size of the largest member.
3.Memory	Each member has its own unique storage location.	All members share the same memory location.
4.Value Altering	Changing one member does not affect other members.	Changing one member alters the value of other members.
5. Initialization	Multiple members can be initialized at once.	Only the first member can be initialized.

Conclusion:

In this experiment, I successfully learned how to define, initialize, and access structures and unions in C++. I understood that a structure is used to store multiple data items of different types, with each member having its own memory space, making it ideal for representing records like employee details. On the other hand, a union allows storing different data types in the same memory location, ensuring memory efficiency but permitting only one active member at a time.

Signature of faculty in-charge with Date: