Unit 7
Structure and Union

Structures

- Arrays allow to define type of variables that can hold several data items of the same kind.
- Whereas structures allows to combine data items of different kinds.

Define Structures

Syntax

```
struct structureName
{
        dataType member1;
        dataType member2;
        ...
};
```

Example

```
struct Person
{
        char name[50];
        int citNo;
        float salary;
};
```

Create struct Variables

- When a struct type is declared, no storage or memory is allocated.
- To allocate memory of a given structure type and work with it, we need to create variables.

Example

```
struct Person {
        // code
};

int main() {
        struct Person person1, person2, p[20];
        return 0;
}
```

OR

```
struct Person {
        // code
} person1, person2, p[20];
```

Access Members of a Structure

- There are two types of operators used for accessing members of a structure.
  1. Dot(.): Member operator
  2. Arrow(->): Structure pointer operator

Example

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

// create struct with person1 variable
struct Person
{
        char name[50];
        int citNo;
        float salary;
};

int main()
{
        struct Person person1;
        // assign values to other person1 variables
        strcpy(person1.name, "Rohan");
        person1.citNo = 1995;
        person1. salary = 2500;

        // print struct variables
        printf("Name: %s\n", person1.name);
        printf("Citizenship No.: %d\n", person1.citNo);
        printf("Salary: %.2f", person1.salary);

        return 0;
}
```

Output

```
Name: Rohan
Citizenship No.: 1995
Salary: 2500.00
```

Keyword typedef

- The typedef keyword is used to create an alias name for data types.
- It is commonly used with structures to simplify the syntax of declaring variables.

Example:

```
struct Distance
{
        int feet;
```

```
        float inch;
};

int main()
{
        struct Distance d1, d2;
}
```

We can use typedef to write an equivalent code with a simplified syntax:

```
typedef struct Distance {
        int feet;
        float inch;
} distances;

int main() {
        distances d1, d2;
}
```

Example

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

// create struct with person1 variable
struct Person
{
        char name[50];
        int citNo;
        float salary;
};
typedef struct Person person;
int main()
{
        person person1;
        // assign values to other person1 variables
        strcpy(person1.name, "Rohan");
        person1.citNo = 1995;
        person1. salary = 2500;

        // print struct variables
        printf("Name: %s\n", person1.name);
        printf("Citizenship No.: %d\n", person1.citNo);
        printf("Salary: %.2f", person1.salary);

        return 0;
}
```

Output:

```
Name: Rohan
Citizenship No.: 1995
Salary: 2500.00
```

Array of structure

- An array having structure as its base type is known as an array of structure.

Example

```
#include<stdio.h>
struct student
{
        char name[30];
        int roll;
        float marks;
};
int main()
{
        /* Declaration of array of structure */
        struct student s[3];
        int i;

        for(i=0;i< 3;i++)
        {
                printf("Enter name, roll and marks of student:\n");
                scanf("%s%d%f",s[i].name, &s[i].roll, &s[i].marks);
        }
        printf("\nInputted details are:\n");
        for(i=0;i< 3;i++)
        {
                printf("Name: %s\n",s[i].name);
                printf("Roll: %d\n", s[i].roll);
                printf("Marks: %0.2f\n\n", s[i].marks);
        }

        return 0;
}
```

Output

```
Enter name, roll and marks of student:
Anu 25 50
Enter name, roll and marks of student:
Gopal 20 66
Enter name, roll and marks of student:
Rabi 30 80
Inputted details are:
Name: Anu
```

```
Roll: 25
Marks: 50.00

Name: Gopal
Roll: 20
Marks: 66.00

Name: Rabi
Roll: 30
Marks: 80.00
```

Passing structure to function

- A structure can be passed to any function from main function or from any sub function.

Example

```
#include <stdio.h>
#include <string.h>
struct Person
{
        char name[50];
        int citNo;
        float salary;
};
void display(struct Person);
int main()
{
        struct Person p;
        strcpy(p.name, "Rohan");
        p.citNo = 1995;
        p. salary = 2500;

        display(p);

        return 0;
}
void display(struct Person p)
{
        // print struct variables
        printf("Name: %s\n", p.name);
        printf("Citizenship No.: %d\n", p.citNo);
        printf("Salary: %.2f", p.salary);
}
```

Output

```
Name: Rohan
Citizenship No.: 1995
Salary: 2500.00
```

Passing array of structure to function

- Array of structure can be passed into user defined function as like built in data types.

Example

```
//C program to pass an arrays of structures to a function
#include<stdio.h>
struct Employee
{
	char name[50];
	int age;
	float salary;
};
// declaration of the function
float calcAverageSalary(struct Employee e[]);
int main()
{
	// array of structure object
	struct Employee e[3];
	int i;
	float avg;
	for(i = 0; i < 3; i++)
	{
		printf("\nEnter name of Employee %d: ", i+1);
		scanf("%s", &e[i].name);
		printf("\nEnter age of Employee %d: ", i+1);
		scanf("%d", &e[i].age);
		printf("\nEnter salary of Employee %d: ", i+1);
		scanf("%f", &e[i].salary);
	}

	// Passing structure to Function
	avg = calcAverageSalary(e);
	printf("\nAverage salary: %f", avg);

	return 0;
}

float calcAverageSalary(struct Employee e[])
{
	float avg;
	int i, sum = 0;
	for(i = 0; i < 3; i++)
	{
		sum += e[i].salary;
```

```
        }
        avg = sum / 3;
        return avg;
}
```

Output

```
Enter name of Employee 1: Ramesh
Enter age of Employee 1: 24
Enter salary of Employee 1: 30000
Enter name of Employee 2: Ganesh
Enter age of Employee 2: 30
Enter salary of Employee 2: 20000
Enter name of Employee 3: Anu
Enter age of Employee 3: 20
Enter salary of Employee 3: 25000
Average salary: 25000.000000
```

Nested Structures

- Structures within a structure is called nested structures.

Example:

```
#include <stdio.h>
struct complex
{
        int imag;
        float real;
};

struct number
{
        struct complex comp;
        int integer;
} num1;

int main()
{
        // initialize complex variables
        num1.comp.imag = 11;
        num1.comp.real = 5.25;

        // initialize number variable
        num1.integer = 6;

        // print struct variables
        printf("Imaginary Part: %d\n", num1.comp.imag);
        printf("Real Part: %.2f\n", num1.comp.real);
        printf("Integer: %d", num1.integer);
```

```
        return 0;
}
```

Output:

```
Imaginary Part: 11
Real Part: 5.25
Integer: 6
```

Union

- A union is a special data type available in C that allows to store different data types in the same memory location.
- A union is a user-defined type similar to structs in C except for one key difference.
- Structures allocate enough space to store all their members, whereas unions can only hold one member value at a time.

Example

```
#include <stdio.h>
union unionJob
{
        //defining a union
        char name[32];
        float salary;
        int workerNo;
} uJob;

struct structJob
{
        char name[32];
        float salary;
        int workerNo;
} sJob;

int main()
{
        printf("size of union = %d bytes", sizeof(uJob));
        printf("\nsize of structure = %d bytes", sizeof(sJob));
        return 0;
}
```

Output

```
size of union = 32 bytes
size of structure = 40 bytes
```

Pointer to structure

- Pointer to structure holds the address of the entire structure.
- It is used to create complex data structures such as linked lists, trees, graphs and so on.
- The members of the structure can be accessed using a special operator called as an arrow operator (->).

Declaration

```
struct tagname *ptr;
```

Accessing

```
ptr-> membername;
```

Example

```
#include <stdio.h>
struct person
{
        int age;
        float weight;
};

int main()
{
        struct person *personPtr, person1;
        personPtr = &person1;

        printf("Enter age: ");
        scanf("%d", &personPtr->age);

        printf("Enter weight: ");
        scanf("%f", &personPtr->weight);

        printf("Displaying:\n");
        printf("Age: %d\n", personPtr->age);
        printf("weight: %f", personPtr->weight);

        return 0;
}
```

Output

```
Enter age: 25
Enter weight: 50
Displaying:
Age: 25
weight: 50.000000
```

Exercise

1. Create a structure “Employee” having Name, Address, Salary, and Age as member functions. Display the name of the employee having aged between 40 and 50 are living in Kathmandu. (5) [TU 2079]
2. Define structure. Explain nested structure with example. Create a structure named book with name, author, and publisher as its members. Write a program using this structure to read data of 50 books and display name of those books published by “XYX” publisher. (10) [TU 2078]
3. Discuss structure of a C Program with suitable example. (5) [TU 2078]
4. What is structure? How is it different from union? Create a structure named course with name, code, and credit_hour as its members. Write a program using this structure to read data of 5courses and display data of those curses with credit_hour greater than 3. (10) [TU 2077]
5. What is structure? Create a structure rectangle with data members length and breadth. (5) [TU 2075]
6. What is structure? How is it different from array? Create a structure student having data members name, roll-number and percentage. Complete the program to display the name of student having percentage greater than or equal to 60. (10) [TU 2074]
7. Explain how structure is different from union? Make a program using structure of booklist having data member‟s title, author, and cost. Enter four data and calculate total cost. (3+4+3) [TU Model]