Contents

Unit 9
Structure and Union

9.0. Structure of the Unit

9.1. Unit Outcomes

After the successful completion of this unit, the student will be able to:

1. Explain the structure declaration and operations on different data types.
2. Apply an array of structures to various applications.
3. Discuss declaration and operations on unions.
4. Develop programs using structure and union.

9.2. Introduction to Structures

A structure is a variable that refers to elements of different data types. Structure overcomes the limitation of an array having a fixed data type. A keyword `struct` is used for defining structure data type. Based on the definition, a structure variable or an array of structures can be created. For example, an entity student includes RollNo, Name, Fees, marks, etc. These fields can be called under the common name student.
The structure definition is given as follows.

```
General Format for Defining Structure:
   struct structure_name {
      data_type member_name1;
      data_type member_name1;
      ....
```

```
        ....
}variable1, varaible2, ...;
Example:
struct student
{
int rollno;
char name[20];
float fees;
};

OR
struct
{
int rollno;
char name[20];
float fees;
}student;
```

Structure variable declaration is done after defining structure members. The structure variables are created as follows.

```
General Syntax:
struct structure_name variable1, variable2, .......;
Example:
struct student s1, s2;
```

Structure members are accessed using a membership operator(.). An example of accessing a structure member is as follows.

```
struct
{
int rollno;
char name[20];
float fees;
}student;
struct student s;
int main(void)
{
printf("enter students records");
scanf(%d %s %f",&s.rollno,&s.name,&s.fees);
printf("the record of a students is…");
printf("%d %s %f",s.rollno,s.name,s.fees);
return 0;
}
```

Program 1: Structure Definition and Structure Variable Declaration

Structure values are initialized using static values as given below.

```
struct student = {1, "vaishali", 12000.00};
```

The assignment of data can be done separately in a `main()` program as follows:

```
  #include <stdio.h>
  struct acno
  {
      int acno;
      float balance;
  };
  struct acno A;
  int main(void)
  {
      A.acno = 12134;
      A.balance = 12000.50;
      printf("Account Number is %d",A.acno);
      printf("\n Balance is %f",A.balance);
      return 0;

}
Sample Input/Output
Account Number is 12134
Balance is 12000.500000
```

Program 3: Constant Data Assignment to Structure Member

`typedef` for structures: The `typedef` keyword is used for data types that already exist. With the use of typedef, the struct keyword can be avoided. The program becomes clearer and more compact. The use of typedef is given below.

```
  #include <stdio.h>
  struct complex
  {
      int real;
      int imag;
  };
  struct complex C;
  int main(void)
  {
      printf("Enter Structure Values");
      scanf("%d %d",&C.real,&C.imag);
      printf("Real Part is %d",C.real);
      printf("\n Imaginary Part is %d",C.imag);
      return 0;
  }
Sample Input/Output
Enter Structure Values 5 6
5 6
```

```
Real Part is 5
Imaginary Part is 6
```

Program3: Structure Program for Complex Numbers

The use of typedef is illustrated in Program 4.

```
  #include <stdio.h>
  #include <string.h>
  typedef struct  {
     char title[50];
     char author[50];
     int book_id;
  } Book;

  int main( ) {
     Book book;
     strcpy( book.title, "C Programming");
     strcpy( book.author, "Vaishali Kulkarni");
     book.book_id = 230413;
     printf( "Book title : %s\n", book.title);
     printf( "Book author : %s\n", book.author);
     printf( "Book book_id : %d\n", book.book_id);

     return 0;
}
Sample Input/Output:
Book title : C Programming
Book author : Vaishali Kulkarni
Book book_id : 230413
```

Program 4: Use of `typedef` in C Program

9.3 Nested Structures

Nested structures refer to the creation of structures within other structures. This means that a structure can contain other structures. It is a way of organizing data in a hierarchical manner, where each level of the hierarchy represents a different level of abstraction. In programming, nested structures are commonly used to represent complex data structures, such as trees and graphs. A program using a nested structure is demonstrated in Program 5.

```
  #include <stdio.h>
  #include <string.h>
  struct Result
  {
  int rollno;
  char studentname[20];
  int marks;
  };
  struct college
  {
  char college_name[20];
  int college_number;
  struct Result R;
  };

  int main()
  {
  // Structure variable
  struct college C;
  strcpy(C.college_name,"GEU");
  C.college_number=55;
  C.R.rollno = 10;
  strcpy(C.R.studentname,"Vaishali");
  printf("College Name : %s\n",C.college_name);
  printf("College Number : %d\n",C.college_number);
  printf("Student id : %d\n",C.R.rollno);
  printf("Student name : %s\n",C.R.studentname);
  return 0;
}
```

Sample Input/Output

```
  College Name : GEU
  College Number : 55
  Student id : 10
Student name : Vaishali
```

Program 5: Nested Structure

An array of Structures: A Structure can be declared as an array of structures for multiple records as given below.

```
  #include <stdio.h>
  #include <string.h>
  struct Result
  {
  int rollno;
  char studentname[20];
  int marks;
  };
  int main()
  {
      int i,n;
  // Structure variable
  struct Result R[10];
  printf("Enter number of Students\n");
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  {
      printf("\n Enter [%d] Record",i);
      scanf("%d %s",&R[i].rollno,&R[i].studentname);
  }
  printf("\n Records are .....\n");
  for(i=1;i<=n;i++)
  {
      printf("%d %s\n",R[i].rollno,R[i].studentname);
  }
  return 0;
}
Sample Input/Output
  Enter the number of Students
  3
  Enter [1] Record
  11 Vaishali
  Enter [2] Record
  23 Gitika
  Enter [3] Record
  36 Ramesh
  Records are .....
  11 Vaishali
  23 Gitika
36 Ramesh
```

Program 6: Array of Structures

9.4. sizeof Structures and Pointer to Structure

The size of a structure in C is the sum of the sizes of its members. In some compilers, padding bits are added, for the alignment of different-sized data members. This is given in Program 7. The data type char takes one byte and the int data type consumes 4 bytes. The total number of bytes is given as 8, which means 3 bytes are added as padding bytes by the compiler.

```
#include <stdio.h>
struct test
{
    int x;
    char y;

};
int main()
{
               printf("Size of struct: %d", sizeof(struct test));
               return 0;
}
Sample Input/Output
Size of struct: 8
```

Program 7 sizeof Structures

Structure and Pointer: A pointer can point to structures and the structure members can be accessed using the arrow operator. This is shown in Program 8.

```
#include <stdio.h>
struct test
{
    int x;
    float y;
  };
  struct test T;

int main()
{

T.x = 10;
T.y = 23.5;
struct test *p;
p = &T;
printf("Data Printed Using Pointer to Structure");
printf(" %d and %f",p->x,p->y);
return 0;
}
```

Sample Input/Output

```
Data Printed Using Pointer to Structure 10 and 23.500000
```

Program 8 Pointer to Structure

A pointer uses an arrow operator for accessing structure members. The address of the entire structure is assigned to a pointer.

9.5. Structure and Functions

A structure can be passed to a function by value or by reference. If there is a requirement, structure members or the entire structure can be passed to a function. Structures can be passed as function arguments like all other data types. Also, a structure member or a pointer to a structure can be returned by a function. C program with structure passing call by value is given below.

```
// Online C compiler to run C program online
#include <stdio.h>
struct TIME
{
    int h,m,s;
};

struct TIME add(struct TIME,struct TIME);
struct TIME add(struct TIME T1, struct TIME T2)
{ struct TIME T3;
    T3.h = T1.h + T2.h;
    T3.h = T3.h + (T1.m + T2.m)/60;
    T3.m = (T1.m + T2.m)%60;
    T3.s = T1.s + T2.s;
    T3.s = T3.s%60;
    return(T3);
}
void print(struct TIME T)
{
    printf("\n Hours: %d  Minutes: %d  Seconds:  %d\n",T.h,T.m,T.s);
}
int main() {
   struct TIME T1 = {2,30,40};
   struct TIME T2 = {3,40,42};
   struct TIME T3 = add(T1,T2);
   print(T1);
   print(T2);
   printf("\n Addition of two time object is:\n");
   print(T3);
   return 0;
}
Sample Input/Output:
Hours: 2  Minutes: 30  Seconds:  40

 Hours: 3  Minutes: 40  Seconds:  42

 Addition of two time object is:

 Hours: 6  Minutes: 10  Seconds:  22
```

Program 9: Pointers as Parameters to Function

9.6 Union

Union data type is like a structure, but it stores all the members of different data types in the same memory location. With his only one union member is accessible at a given time. The general syntax of union is as follows:

```
General Syntax:
union union_name {
   datatype member1;
   datatype member2;
   ...
};
union student
{
  int rollno;
float fees;
}S;
```

Union variables can be declared by two methods 1) Immediately after creating the union data type and 2) By giving the union name and a list of variables. This is illustrated as follows.

```
1)
 union union_name
{
 datatype member1;
 datatype member2;
 ...
} variable1, variable2, ...;

2) union union_name variable1, variable2, variable3...;
```

Union variable members are accessed using a dot (.) operator as given below.

```
variable1.member1;
```

A program using a union variable is shown in Program 10.

```
// C Program to demonstrate how to use union
#include <stdio.h>

// union declaration
union test{
               int x;
               char y;
               float z;
};


int main()
{

               // defining a union variable
               union test v1;

               //Initializing the union member
               v1.x = 15;

               printf("The value stored in member1 = %d ",v1.x);
    v1.y = 'Y';
    printf("The value stored in member2 = %c",v1.y);
               return 0;
}
Sample Input/Output
The value stored in member1 = 15 The value stored in member2 = Y
```

Program 10:Union Data Type

In the above program at a given instance, either x, y, or z variables can be used. All three members cannot be used simultaneously. The size of the union is determined by the size of the largest member of the union. Sizeof union is demonstrated in Program 11.

```
#include <stdio.h>
union test {
```

```
                int a[10];
                char y;
} Test;

// driver code
int main()
{
                // finding size using sizeof() operator

                int size = sizeof(Test);

                printf("Sizeof test: %d\n", size);
                return 0;
}
Sample Input/Output:
Sizeof test: 40
```

Program11: sizeof union

Union is useful in applications when multiple data type elements need to be stored in the same memory location. Union data types can be used in a nested form.

9.7. Structure and Union

Structures and unions are user-defined data types used for storing different data types using a common name. Both data types can be used for creating arrays. The assignment and sizeof operators work the same for both the data types. The parameter passing mechanisms are the same for structure and union: call by value and call by reference. Structure and union members are accessed using the dot operator (.).

The main difference between structure and union is the availability of data. In structure, all the members of different data types can be used at a given instance. In union, though there are multiple data types, only one can be used at any given instance. The size of memory for a structure is the sum of the sizes required for all the data members. In the union data type, the size is the memory requirement is given by the size of the largest number. In structure, multiple members can be initialized at a time, whereas in the case of a union, only one member can be accessed and initialized.

9.8 Self-Assessment Questions/Activities

Q1. Explain the general syntax for defining and declaring a structure. Discuss with an example. (5 Marks, L1)

Q2. What is the syntax for defining and declaring a union type? Discuss with an example. (5 Marks, L1)

Q3. Compare union and structure data types. (5 Marks, L1)

Q4. Write a C program to create a bank account. Write a function that finds the account with maximum balance using structure variables (10 Marks, L2)

Q5. Write a C program to declare a union data type for printing days from Monday to Sunday. Use a function print() that reads a union variable for printing any one day. (10 Marks, L2)

Q6. Write a C program using structures and functions for the following operations. (10 Marks, L3)

1. Read student’s records as marks in three subjects.
2. Find the average and the student with the highest balance.

Q7. Write a C program to add two objects of rectangular coordinates. Write a function to convert rectangular coordinates into polar form. (10 Marks, L3)

Q8. Write a C program to calculate the salary of an employee using fields name, basic salary, allowances, deductions and print the net salary. Use the concept of an array of structures. (10 Marks, L3)

9.9 Multiple-Choice Questions (MCQs)

Q1. A collection of different data types under a common name is____(1 Mark, L1)

A. String
B. Structure
C. Array
D. None of the above

Q2. Members of the union are accessed using the______ operator. (1 Mark, L1)

A. Array
B. &
C. Dot (.)
D. Asterisk (*)

Q3. _____ of the following cannot be a member of a structure.

A. Array
B. Pointer
C. Structure
D. None

Q4. The size of the union in the following program is______ (1 Mark, L2)

```
union test
{ int x;
   float y;
   char z;
}
```

A. 10 Bytes
B. 7 Bytes
C. 12 Bytes
D. 5 Bytes

Q5. The size of the union is size of the largest element. State True or False (1 Mark, L1)

A. False
B. True

9.10. Keys to Multiple Choice Questions

Q1. Structure (B)
Q2. Dot (.) (C)
Q3. None (D)
Q4. 10 Bytes
Q5. True (B)

9.11 Summary of the Unit

Structure and union are user-defined data types that are used for saving variables of different data types using a common name. In a union, only one member can be accessed and initialized at a time. Structure and union need to be declared and initialized before creating a structure variable. The structure is commonly used for creating records. An array of structures is used for storing databases. Structure and union include members of different data types. These members are accessed using the dot operator (.). Structure and union can be accessed using pointers. One structure can contain another structure, this is known as nested structure. For declaring structure, a struct keyword is used. To avoid using struct keyword, typedef can be used.

9.12. Keywords

- Structure
- Union
- Array of structure
- Sizeof structure
- Nested structure
- Union

9.13 Recommended Reading

Herbert Schildt. (2017). C Programming: The Complete Reference. 4th ed. USA: McGraw-Hill.