UNIT:5

STRUCTURE & UNION

Topics to be covered…

- 5.1 Introduction and features of structure
- 5.2 Declaration and Initialization of structure
- 5.3 Array of structure
- 5.4 Pointers to structure
- 5.5 Typedef
- 5.6 Enumarated Data type
- 5.7 Union
- 5.8 Union of Structures

STRUCTURE

- Structure means to represents a collection of data items of different types using a single name
- **Syntax for defining the structure is**:

```
struct  struct_name
{
        data type var2;
        …..
};
```

- The keyword struct is used to define a structure template.
- Struct_name is an optional tag which provide name of structure.

- It is user defined. The member of structure is written in curly{ } braces.

- **Example:**

  Time: seconds, minutes, hours

  Date: day, month, year

  Address: name, door number, city,taluka, dist.
- The structure definition is terminated by semicolon(;) symbol.
- **Example:**

```
Struct book_bank
{
char title[20];
char author[15];
int  pages;
float prices;
};
```

STRUCTURE IS INITIALIZE

- We can also initialize the structure variable at the time of creation of Variables.
- **For example**:

```
                struct student
                    {
                      int roll_no;
                       char name[20];
                           int age;
                    };
                        .............
                        ..............
                          struct student s1={1,"vansh",23};
```

- This will make following initialization,
- **s1.roll_no=1,s1.name="vansh". S1.age=23**

- If all the value are not supplied but partial initialization is made,
- Then rest will be initialized as zero for number and NULL to strings.
- <u>**For Example**</u>:
- struct student s1={a,"vansh"};
- will assign 1 to s1.roll_no and "vansh" to s1.nameand third value is zero.

DIFFERENCE BETWEEN ARRAY AND STRUCTURE

ARRAY	STRUCTURE
An array is collection of related data of same data type.	Structure is a collection of related data of different types.
An array is derived data type.	Structure is a user defined one.

EXAMPLE:
INT A[5];
	EXAMPLE
	STRUCT STUDENT
	{
	INT S_NAME[5];
	INT S_ENROLL;
	};

HOW THE MEMBERS OF STRUCTURE CAN BE ACCESSED

- We can access the individual member of structure using dot(.) Operator.
- **Syntax is**:

  struct_var.struct_member
- Where struct_var is a variable of structure type,while struct_member is a name of member variable of structure.
- For **example,** the structure student with s1 & s2 as variables.

```
s1.roll_no=5;
                    s1.age=20;
                    s2.roll_no=7;
                    s2.age=21;
```

- Here, code is assign student s1 to roll_no=5 and age=20.
- While for student s2 to roll_no=7 and age=21.

Array of structure

- An array of structure, each element of the array representing a structure variable.
- For example: struct student s[50];
- It defines an array s[50] that consists of 50 elements.
- Following ex. shows how an array of structure can be declared:
-

```
                    Struct marks
                        {
                          int x;
                          int y;
                        };
                      void  main()
                       {
                          struct marks[3]={{5,7},{7,9},{9,10}};
```

- An array actually look as shown in fig.

```
S[0].x=5;
S[0].y=7;
S[1].x=7;
S[1].y=9;
S[2].x=9;
S[2].y=10;
```

POINTER TO STRUCTURE

- For representing complex data structures,we can use structures and pointers together.
- Pointer can be member of structure,or pointer to structure,or an array of pointers to str.
- **For Example:**

```
struct student
      {
        int roll_no;
        char name[20];
        char address[30];
        int age;
       }s1,*sptr;
       ……….
       ………..
       sptr=&s1;
```

- Now,sptr points to s1 and can be used to access member variables of struct student.
- To access member variables using pointer to structure,we have to use an arrow -> operator.
- **The Syntax is:**
- **struct_pointer->member_var;**
- The struct_pointer is a pointer to structure, and member_var is the name of member variable.
- **For Example:**
- **S1->roll_no=20;**
- We can also use dot(.)operator to access member variable like:
- **(*s1).roll_no=20;**
- Both above statements do the same things.

TYPEDEF:

- The C programming language provide a keyword called **typedef**
- **Example:-** one byte numbers
- Typedef unsigned char BYTE;
- #include<stdio.h>
- #include<string.h>
- #include<conio.h>

- typedef struct Books
- {
- char title[50];
- char author[50];
- Char subject[100];
- Int book_id;
- }book;
- Void main()
- {
- Book book;
- Strcpy(book.title,"c programming");
- Strcpy(book.author,"xyz");
- Strcpy(book.subject,"acp");
- Book.book_id=6495407;
- Printf("book title:%s\n",book.title);
- Printf("book author:%s\n",book.author);
- Printf(book subject:%s\n",book.subject);
- Printf("book book_id:%d\n",book.book_id);
- Getch();
- }

ENUMERATED DATA TYPE

- Enum is known as a “Enumerated Data Type”.
- Keyword enum is used to defined enumerated data type.
- It is user defined data type.
- SYNTAX:
- Enum identifier{value1,value2,….value n};
- Example:
- Void main()
- {
- Int I;
- Enum month{JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,DEC};
- Clrscr();
- For(i=JAN;i<=DEC;i++)
- {
- Printf(“\n%d”,i);
- }
- Getch();
- }

UNION

- It is user defined data type like structure.
- **Major difference between them:**
- **in structure**, each member has its own storage location.so,it occupies more memory.
- **In union** , all the member of union use the same location. so, it occupies less memory.
- **Syntax is:**

```
Union union name
      {
        data type mem1;
        data type mem2;
        ………….
        …………
      };
```

- **For Example:**

```
                    union item
                     {
                         int m;
                         float x;
                         char c;
                      };
```

- The union contain three memory, each with different data type. We can use one of them at a time.
- Show memory allocation for union:

1000	1001	1002	1003

```
  <---char ---->
 <-------------int------------>
  <-------------------------------float----------------→
```

DIFFERENCE BETWEEN STRUCTURE AND UNION

Structure	Union
➢ Each member has its own separate storage space.	➢ all member shares the storage space.
➢ Memory occupied is the total required to store all the members.	➢ Memory occupied is only that much which is required to store the largest member of union..

➢ Ex:
➢ struct emp
{
int eid;
char name[20];
int age;
float salary;
};
Total size required=
2+20+2+4=28 bytes
	➢ Ex:
	➢ union emp
	{
	int eid;
	char name[20];
	int age;
	float salary;
	};
	Total memory required=
	max(2,20,2,4)=20 bytes

IMP QUESTION

1. Define Structure.
2. Define declaration of Structure & Initialization of Structure.
3. Give differences between Array & Structure.
4. Define a structure name "customer" which has the following fields.

(i)Account no	-a positive integer.
(ii)Name	-a string of 25 character.
(iii)Balance	-a real values.
(iv) Account type	-a character.

5. Explain Union. Explain with Example.
6. Give differences between Structure & Union.

7. Explain array of structure using suitable example.
8. Explain nesting of structures using example.

9. Give structure definition to represent Cricketer.

10. How to declare and initialized structure?

11. Explain array of structure with example.