Chapter 16

By St Joseph

WORD COUNT	2489	TIME SUBMITTED	09-JUL-2016 11:26PM
		PAPER ID	23989571

16. STRUCTURES & UNIONS

16. 1 STRUCTURES:

A Structure is a collection of one or more variables of different data types, grouped together under a single name. By using structures, we can make a group of variables, arrays, pointers etc.

Features of Structures:

- It is possible to copy the contents of all structure elements of different data types to another structure variable of its type using assignment operator.
- Nesting of structure is possible.
- It is possible to pass structure elements to a function.
- It is possible to create structure pointers.

16.2 Declaration & Initialization of Structures:

Structures can be declared as follows:

```
struct struct_name
{
        datatype variable1;
        datatype variable2;
};


struct struct_name variable1,variable2;
```

Accessing Structure Members:

The structure member is accessed using *member operator* "." Which is also known as 'dot operator'.

Three ways to access members:

- Using dot notation : v.x
- Using indirection notation : (*ptr).x
- Using selection notation : ptr->x

Example:

Creating a structure for Student record

```
struct student
{          int rollno;
           char name[25];
           int marks[6];
           int avg;
 }
struct student s1;
```

Example program:

```
#include <stdio.h>
void main()
{
struct student
{
int regno;
char name[30];
char branch[30];
int marks[10];
};
struct student s;
int total=0,i,n;
clrscr();
printf("Enter the reg.No");
scanf("%d",&s.regno);
fflush(stdin);
printf("\nEnter the name");
gets(s.name);
fflush(stdin);
printf("\nEnter the branch");
gets(s.branch);
printf("\nEnter the marks one by one\n");
for(i=0;i<7;i++)
{
scanf("%d",&s.marks[i]);
total=total+s.marks[i];
}
clrscr();
printf("\n\n_______STUDENT MARK PROCESSING_______________\n");
printf("\n\t Register No          :  %d",s.regno);
printf("\n\t Name of the Student  :  %s",s.name);
printf("\n\t Branch               :  %s",s.branch);
printf("\n\t  English              :  %d",s.marks[0]);
printf("\n\t  Mathematics          :  %d",s.marks[1]);
printf("\n\t  Chemistry            :  %d",s.marks[2]);
printf("\n\t  Physics              :  %d",s.marks[3]);
printf("\n\t  Computer Programming  :  %d",s.marks[4]);
printf("\n\t Electronic Devices          :  %d",s.marks[5]);
if(s.marks[0]>=50&&s.marks[1]>=50&&s.marks[2]>=50&&s.marks[3]>=50&&
s.marks[4]>=50&&s.marks[5]>=50&&s.marks[6]>=50)
printf("\n\t  Result             :  PASS");
else
printf("\n\t  Result                 :  FAIL");
printf("\n\t  Total                  :  %d",total);
getch();
}
```

OUTPUT:

```
Enter the reg.No101
Enter the nameaarthi
Enter the branchcse
Enter the marks one by one
89
90
99
```

```
98

99

88
____________STUDENT MARK PROCESSING_______________

        Register No            :  101
        Name of the Student    :  aarthi
        Branch            :   cse
         English                :  89
         Mathematics       :  90
         Chemistry                :  99
         Physics                :  98
         Computer Programming  :  99
        Electronic Devices        :  88
         Result               :  PASS
         Total            :  641
```

16.3 Arrays of Structures:

Arrays of structures are collection of structures which is used to store different type of structure member variables. Array of structure is used to handle more records within one structure.

Example:

```
struct book
{
 char name[10];
 int price;
};
struct book b[3];
void main( )
{
  int i;
  for(i=1;i<=3;i++)
   {
      Printf("Enter the book name, price:\n");
      Scanf("%s%d",&b[i].name,&b[i].price);
```

```
          }
          for(i=1;i<=3;i++)
          printf("\n%s\t %d",b[i].name,b[i].price);
          getch( );
     }
```

OUTPUT:

```
Enter the book name,price:
English  165
Enter the book name,price:
Maths  250
Enter the book name,price:
Physics  193
```

16.4 STRUCTURES WITHIN STRUCTURES:

One structure declared inside other structure is called as **structures within a structure** otherwise known as ***nesting* of structures.** We can write one Structure inside another structure as **member** of another structure. The structure variables may be a normal structure variable or a pointer variable to access the data in the structure.

Syntax:

```
   Struct structure_name1
  {
  datatype declarations;
  };
   Struct structure_name2
   {
   declarations;
   ……………………
   ……………………
   struct structure_name1;
   variable_name1;
  …………………
  …………………
 };
```

Example:

```
#include<stdio.h>
main( )
{
 struct date
{
  int day;
 char month[20];
 int year;
};
 ttruct employee
  {
   int code;
   char name[30];
   float salary;
  struct date doj;
  };
 struct employee emp1;
 printf("\n Enter  Employee code:");
 scanf("%d",&emp1.code);
 printf("\n Enter  Employee name:");
 scanf("%s",&emp1.name);
 printf("\n Enter  Employee salary:");
scanf("%f",&emp1.salary);
printf("\n Enter  Employee date of  joining:");
scanf("%d %s %d",&emp1.doj.day,&emp1.doj.month,&emp1.doj.year);
printf("\n The Employee code is %d",emp1.code);
printf("\n The Employee Name is %s",emp1.name);
printf("\n The Employee Salary is %f",emp1.salary);
printf("\n The Employee DOJ %d %s %d",emp1.doj.day,emp1.doj.month,emp1.doj.year);
}
```

INPUT:

```
Enter Employee code:200
Enter Employee name: Ram
Enter Employee salary:22000.0
Enter Employee Doj is:12 December 2014
```

OUTPUT:

```
The Employee Code is 200
The Employee Name is Ram
The Employee Salary is22000.0
The Employee DOJ is 12 December 2014
```

16.5 STRUCTURES AND FUNCTION

Three methods by which the values of a structure can be transferred from one function to another function.

- Pass each member of the structure as an actual argument of the function.
- Pass a copy of the entire structure to the called function.
- Pass the address location of the structure to the called function.

Syntax:

```
function_name(structure_variable_name);
datatype function_name(struct_type st_name)
{
……………………..
……………………..
return(expression);
}
```

Example:

/*Passing a copy of entire structure to a function*/

```
struct std
 {
  int no;
  float avg;
  };
  void fun(struct std p);
  void main( )
  {
    struct std a;
   clrscr( );
   a.no=1;
   a.avg=90.6;
   fun(a);
   getch( );
  }
  void fun(struct std p)
 {
  printf("Number is:%d\n",p.no);
 printf("Average is:%f\n",p.avg);
 }
```

Output:

```
Number is  : 1
Average is  : 90.599998
```

Structure and pointers

A structure containing a member that is a pointer to the same structure type is called self referential structure.

Syntax:

```
                    struct tag_name
                   {
                   datatype declarations;
                   }*structure_name;
```

"->" is called as structure pointer symbol.

Example:

Structure using pointers

```
#include<stdio.h>
void main( )
{
     struct
    {
      int rollno;
     char name[30];
     char branch[4];
     int marks;
  }*stud;
clrscr();
printf("\n enter the rollno:");
scanf("%d",&stud->rollno);
printf("\n enter name:");
scanf("%s",stud->name);
printf("%s",enter the branch:");
printf("\n Roll Number:%d",stud->rollno);
printf("\n Name:%s",stud->name);
printf("\n Branch:%s",stud->branch);
getch();
}
```

Output:

```
Enter rollno   : 1001
Enter name     :  Raj
Enter branch   : CSE


Roll Number  : 1001
Name           :  Raj
Branch         :  CSE
```

Array and structure has similar properties. The name array indirectly specifies the address of its zeroth element. Similarly the name of the structure specifies the address of the zeroth element in the zeroth record.

Example

```
struct item
{  char name[20];
    int ID;
    float price;
} product[5], *ptr;
```

The above segment declares product as an array of 5 elements, each of the type **struct item** and **ptr** is a pointer to the data object of the type **struct item.**

The assignment   ptr = product; assigns the address of the zeroth element of **product** to **ptr.**

Now ptr will point to product[0]. All the structure members name. ID, and price can be accessed through pointer as follows.

```
ptr -> name
ptr->ID
ptr ->price.
```

The symbol **->** arrow operator is also knows as member selection operator. This special operator is made uo of a minus sign and greater than symbol. Each time when the pointer ptr is incremented by one, it automatically points to the next record. The scale factor depends on the size of the structure elements.

To access members of structure with structure variable, we used the dot . operator. But when we have a pointer of structure type, we use arrow -> to access structure members.

Example program

```
Struct book
{
      char name[20];
       float price;
```

```
                }
     void main()
     {
      struct book b;
      struct book *ptr = &b;
      ptr->name = “William Stallings”;
      ptr->price = 250;
    }
```

Example program

```
#include<stdio.h>
#include<string.h>
Struct student
                {  char name[25];
                     int id;
                     float avg;
                };
void main()
{ int i;
   struct student rec1 = “ Aswin”,1,90.5}
   struct student *ptr;
    ptr = &rec1;
    printf(“ Record of the Student 1 : \n”);
    printf(“\n Name of the student = %s \n”, ptr->name);
    printf(“\n Id   is = %d”, ptr->id);
   printf(“\n Average Mark = %f\n”, ptr->avg);
```

```
}
```

Output

```
Record of the Student 1 :

Name of the student = Aswin

Id   is = 1

Average Mark = 90.50000
```

16.6 DIFFERENCE BETWEEN ARRAYS AND STRUCTURES

Sl.no	ARRAYS	STRUCTURES
1	Single name that represent a collection of data items of same data type.	It is a single name that represents a collection of data items of different data types.
2	Individual data in an array are called elements.	Individual data in a structure are called members.
3	There is no keyword to represent arrays,but the square braces[] preceding the variable name tells us that we are dealing with array.	The keyword struct tells us that we are dealing with structure.
4	The array elements are accessed by its name followed by square brackets[] within which the subscript value is placed.	The members of a structure are accessed by the dot operator.

Example:

```
#include <stdio.h>
typedef struct complex{
  float real;
  float imag;
}complex;
complex add(complex n1,complex n2);
int main(){
    complex n1,n2,temp;
    printf("For 1st complex number \n");
    printf("Enter real and imaginary respectively:\n");
    scanf("%f%f",&n1.real,&n1.imag);
    printf("\nFor 2nd complex number \n");
    printf("Enter real and imaginary respectively:\n");
    scanf("%f%f",&n2.real,&n2.imag);
    temp=add(n1,n2);
    printf("Sum=%.1f+%.1fi",temp.real,temp.imag);
    return 0;
}
complex add(complex n1,complex n2){
      complex temp;
      temp.real=n1.real+n2.real;
      temp.imag=n1.imag+n2.imag;
      return(temp);
}
```

Output:

```
For 1st complex number
Enter real and imaginary respectively: 2.3
4.5

For 1st complex number
Enter real and imaginary respectively: 3.4
5
Sum=5.7+9.5i
```

16.7 UNION:

Union is a variable, which is similar to the structure. It contains data elements of different data types. The Union requires bytes that are equal to the number of bytes required for the largest members.

The syntax is as follows:

```
union result
{
int marks;
int grade;
}u1;
```

Here the memory allocation is 2 bytes, whereas if it is a structure the memory allocation is 4 bytes.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
union student
{
char name[20];
int rollno;
char branch[10];
}student;
clrscr();
printf("enter student name,roll number and branch");
scanf("%s%d%s",student.name,&student.rollno,student.branch);
printf("\n student name:%s",student.name);
printf("\n student rollno:%d",student.rollno);
printf("\n student branch:%s",student.branch);
getch();
}
```

Output:

```
Enter student name,roll number and branch
Angel
1560
CSE
Student name:Angel
Student rollno:1560
```

Student Branch:CSE

16.8 Difference between union and structure:

Though unions are similar to structure in so many ways, the difference between them is crucial to understand. This can be demonstrated by this example:

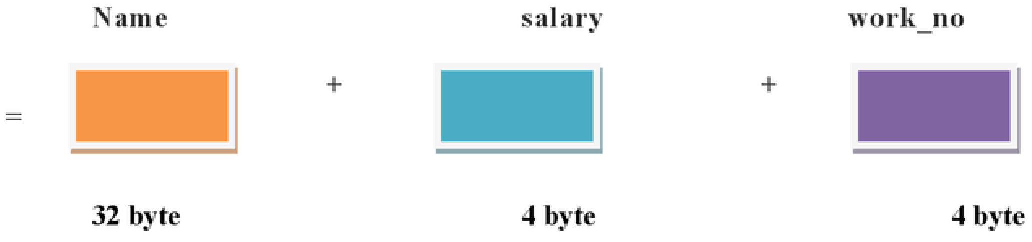
Example:

```
#include <stdio.h>
union job
{
   char name[32];
   float salary;
   int worker_no;
}u;
struct job1  {
   char name[32];
   float salary;
   int worker_no;
}s;
int main(){
   printf("size of union = %d",sizeof(u));
   printf("\n size of structure = %d", sizeof(s));
   return 0;
}
```

Output:

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

Structure memory allocation:



Union memory allocation:

name

Difference between Structure & Union

S.No	Structure	Union
1.	The keyword is struct.	The keyword is union.

2.	Memory allocation is done for all the data members in the structures.
	Example.
	struct student
	{
	int rollno;
	char name[5];
	}s1;
	The memory allocation is 7 Bytes.
		Memory allocation is done for the data member which requires maximum allocations.
		Example:
		union student
		{
		int rollno;
		char name[5];
		}
		The memory allocation is 5 Bytes

3.	All the data members are available in the primary memory at any time of execution.	Only the last stored data element is available in the primary memory at any time of execution.
4.	Since memory is allocated for all the data members, no data is deleted in the primary memory	Since memory is not allocated for all the data member, only one data is available and other data is deleted from the primary memory

16.9 Passing Union to a Function:

Union can be passed as structures in C programming.

- Passing by value (passing actual value as argument).
- Passing by reference (passing address as argument).

Passing structure by value:

A structure variable can be passed to the function as an argument as normal variable. If structure is passed by value, change made in structure variable in function definition does not reflect in original structure variable in calling function.

Example:

```
#include<stdio.h>
struct student{
char name[50];
int rollno;
};
void display(struct student stu);
int main( )
{
struct student s1;
printf(“\n enter the name:”);
scanf(“%s”,s1.name);
printf(“\n enter the roll no:”);
scanf(“%d’,s1.rollno);
display(s1);
return 0;
}
Void display(struct student stu)
{
Printf(“\n Name:%s”,stu.name);
Printf(“\n Roll no:%d”,stu.rollno);
}
```

Output:

```
Enter student name:Max
Enter rollno:101

Name:Max
Roll no:101
```

Passing structure by reference:

The address location of structure variable is passed to function while passing it by reference. If structure is passed by reference, change made in structure variable in function definition reflects in original structure variable in the calling function.

Example:

```
#include<stdio.h>
struct distance
{
    int feet;
    float inch;
};
void Add(struct distance d1,struct distance d2, struct distance *d3);

int main()
{
    struct distance dist1, dist2, dist3;
    printf("First distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist1.feet);
    printf("Enter inch: ");
    scanf("%f",&dist1.inch);
    printf("Second distance\n");
    printf("Enter feet: ");
    scanf("%d",&dist2.feet);
    printf("Enter inch: ");
    scanf("%f",&dist2.inch);
    Add(dist1, dist2, &dist3);
    printf("\nSum of distances = %d\'-   %.1f\"",dist3.feet, dist3.inch);
    return 0;
}
void Add(struct distance d1,struct distance d2, struct distance *d3)
{
     d3->feet=d1.feet+d2.feet;
     d3->inch=d1.inch+d2.inch;
     if (d3->inch>=12)
{
         d3->inch-=12;
         ++d3->feet;
    }
}
```

Output:

```
First distance
```

```
Enter feet: 12
Enter inch: 6.8
Second distance
Enter feet: 5
Enter inch: 7.5

Sum of distances = 18'-2.3"
```

What difference does it make between structure and union?

All members of structure can be accessed at any time. But, only one member of union can be accessed at a time in case of union and other members will contain **garbage** value.

Additional Example program:

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

union item
{
 int a;
 float b;
 char ch;
};

int main( )
{
 union item it;
 it.a = 12;
 it.b = 20.2;
 it.ch='z';
 clrscr();
 printf("%d\n",it.a);
 printf("%f\n",it.b);
 printf("%c\n",it.ch);
 getch();
 return 0;
}
```

Output:

```
-26426
20.1999
z
```

16. 10 Pointer to union :

Pointer which stores address of union is called as pointer to union

Syntax:

```
union team t1;
union team *sptr;
sptr = &t1;
```

Example:

```
#include<stdio.h>

union team {
  char *name;
  int members;
  char captain[20];
};

int main()
{
union team t1,*sptr = &t1;

t1.name = "India";
printf("\nTeam : %s",(*sptr).name);
printf("\nTeam : %s",sptr->name);

return 0;
}
```

Output:

```
Team = India
Team = India
```

Program Explanation:

- **sptr** is **pointer to union** address.
- -> and **(*).** both represents the same.
- These operators are used to access data member of structure by using **union's pointer**.

Chapter 16

ORIGINALITY REPORT

50%

SIMILARITY INDEX

PRIMARY SOURCES

1	www.whitehatadda.in Internet	315 words — 12%
2	www.programiz.com Internet	130 words — 5%
3	www.c4learn.com Internet	89 words — 4%
4	www.slideshare.net Internet	86 words — 3%
5	fr.slideshare.net Internet	58 words — 2%
6	www.docstoc.com Internet	57 words — 2%
7	www.studytonight.com Internet	53 words — 2%
8	learnclanguage.com Internet	52 words — 2%
9	www.expertsmind.com Internet	47 words — 2%
10	jntuportal.com Internet	44 words — 2%
11	fastlearned.blogspot.com Internet	43 words — 2%

12	www.giftofnepal.com Internet	39 words — 2%
13	srividyaengg.ac.in Internet	30 words — 1%
14	www.vvit.in Internet	29 words — 1%
15	m.blog.csdn.net Internet	22 words — 1%
16	ppitcse.blogspot.com Internet	21 words — 1%
17	www.cprogrammingexpert.com Internet	18 words — 1%
18	fresh2refresh.com Internet	17 words — 1%
19	www.authorstream.com Internet	16 words — 1%
20	wwwcs.dongguk.ac.kr Internet	15 words — 1%
21	www.winu.cn Internet	13 words — 1%
22	pcgurulive.com Internet	13 words — 1%
23	yuliana.lecturer.pens.ac.id Internet	12 words — < 1%
24	www.sk4education.com Internet	9 words — < 1%
25	dpacslayer.blogspot.com Internet	9 words — < 1%

26	archive.mu.ac.in Internet	9 words — < 1%
27	clanguagelearnonline.blogspot.com Internet	8 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		