Date
Page

Structure -

Array allow you to define type of variables that can hold several data items of the same kind but structure is another user defined data type available in C programming, which allows you to combine data items of different kind.

It is the collection of dissimilar data types or hetrogenous data types grouped together.

Each individual data item within the structure is referred to as member.

A structure definition is specified by using the keyword 'struct'.

* Structure declaration -

```
struct student /*Definition of structure*/
{ int roll;
  float marks;   /* members */
  char name[10];
};
void main()
{
  struct student S1;  /* Variable of the structure student */
}
```

(or)

```
struct student
{ int roll;
  float marks;
  char name[10];
} S1, S2;
```

* Members of structure are allowed space in the memory only when the objects are declared (Eg- S1, S2)

Date
Page

* In structure, for using the members, we use a dot or period operator(.).

Eg:

```
#include <stdio.h>
struct student
{ int rollno;
  char name[20];
  float marks;
};
Void main()
{ struct student S1 = {1, "Abhi", 90};
  struct student S2 = {2, "Ak", 85};
  printf("Information of S1");
printf("\n %d %s %f", S1.rollno, S1.name,
                      S1.marks);
  printf("Information of S2");
printf("\n %d %s %f", S2.rollno, S2.name,
                      S2.marks};
}
```

* Reading information of student using scanf :-

```
printf("Enter info. for S1");
scanf("%d, %s %f", &S1.rollno, &S1.name,
                   &S1.marks);
```

* Array of structure -

When database of any element is used in huge amount, we prefer array of structure

Eg:-

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

struct student
{
    char name[30];
    char branch[25];
    int roll;
};
void main()
{
  struct student s[200];
   int i;
   s[i].roll = i+1;
  printf("\n Enter information of students");
    for(i=0; i<200; i++)
     {
       printf("\n Enter the roll no. %d\n", s[i].roll);
      printf("\n Enter the name:");
     scanf("%s", s[i].name);
    printf("Enter the branch");
     scanf("%s", s[i].branch);
     printf("\n");
    }
  printf("\n Displaying information of students:\n\n");
    for(i=0; i<200; i++){
     {
       printf("\n\nInformation for roll no %d:\n", i+1);
```

```
        printf ("\n Name : ");
        puts (S[i].name);
        printf ("\n Branch : ");
        puts (S[i].branch);
    }
}
```

* Array within structure -

```
#include <stdio.h>
struct student
{   char name[15];
    int m[3];  // marks
    int total;
};
void main()
{   struct student s[3];
    int i, j;
    for (i=0; i<3; i++)
    {   printf ("Enter the name : ");
        scanf ("%s", S[i].name);
        S[i].total = 0;
        for (j=0; j<3; j++)
        {   printf ("Input marks ");
            scanf ("%d", S[i].m[j]);
            S[i].total = S[i].total + S[i].m[j];
        }
    }
    printf ("\n Result of students are as");
    for (i=0; i<3; i++)
    {   printf ("%s", S[i].name);
```

```
for (j=0; j<3; j++)
  { printf ("%d", S[i].m[j]);
    printf ("%d", S[i].total);
```

* Nested structure -

```
#include <stdio.h>
struct student {
{  char name[30];
   int roll;
   struct dateOfBirth
    {  int dd;
       int mm;
       int yy;
    } DOB;
};
int main()
{
   struct student std;
   printf("Enter name:");
    gets(std.name);
   printf("Enter roll no:");
    scanf("%d", &std.roll);
   printf("Enter date of Birth[DDMMYY] format:");
   scanf("%d %d %d", &std.DOB.dd, &std.DOB.mm, &std.DOB.yy);
   printf("\n name : %s", std.name);
    printf("\n roll: %d", std.roll);
    printf("\n Date of birth: %.02d/%.02d/%.02d",
                    std.DOB.dd, std.DOB.mm, std.DOB.yy);

   return 0;
}
```

* Show how a structure variable is passed as a parameter to a function, with an example.

In C, structure can be passed to function by two methods -

i) Pass by value
(Passing actual value of structure variable as argum

ii) Pass by reference
(Passing address of structure variable as argum

* Passing structure using call by value -

```
#include <stdio.h>
struct student
{
   char name[50];
    int roll;
};

void display(struct student stu);
int main()
{
    struct student s1;
    printf("Enter student's name");
    scanf("%s", s1.name);
    printf("Enter roll no.");
     scanf("%d", &s1.roll);
     display(s1);

}
```

function prototype should be below to the structure declaration otherwise compiler shows error

```
void display(struct student stu)
  {
     printf("Entered Name: %s", stu.name);
     printf("\n Roll no: %d", stu.roll no);
  }
```

* Passing structure using call by reference -

```
#include <stdio.h>
struct student
 {
    char name[50]
     int roll;
 };
void display( struct student *stu);
int main()
  {
     struct student s1 = {"Abhi", 32};

        display (&s1);
  }

void display ( struct student * stu)
  {
     printf("Student's name = %s \n", stu->name);
      printf(" Student's rollno = %d \n", stu->roll);
  }
```

Date
Page

* Structure with pointer -

```
#include <stdio.h>
struct item
{
    char itemName[30];
     int qty;
     float price;
     float amount;
};

int main()
{
     struct item itm;
     struct item *pitem;
          pitem = &itm;
printf ("Enter product name: ");
   gets (pitem -> itemName);
printf ("Enter price: ");
   scanf ("%f", &pitem -> price);
printf ("Enter quantity: ");
   scanf ("%d", &pitem -> qty);

   pitem -> amount = (float) pitem -> qty * pitem -> pric

printf ("\n Name : %s", pitem -> itemName);
printf ("\n Price: %f", pitem -> price);
printf ("\n Quantity : %d", pitem -> qty);
printf ("\n Total amount : %f", pitem -> amount);

}
```

* Structure pointer using user define function -

```
# include <stdio.h>

struct item
 {
    char itemName[30];
    int qty;
    float price;
    float amount;
};

void readItem (struct item *i)
 {
    printf ("Enter product name:");
     gets (i->itemName);
     printf ("Enter price");
         scanf ("%f", &i->price);
    printf (" Enter quantity : ");
     scanf ("%d", &i->qty);

    i->amount = (float)i->qty * i->price;
 }

Void printItem (struct item *i)
    printf ("\n Name: %s", i->itemName);
    printf ("\n Price : %f", i->price);
    printf ("\n Quantity : %d", i->qty);
    printf ("\n Total amount : %f", i->amount);
int main()
{   struct item itm, *pitem;
      pitem = &itm;
     readItem (pitem);
     printItem (pitem);
}
```

Date
Page

* Type definition (typedef) :-

The typedef is a keyword using which the programmer can create a new data type name for an already existing data type name. So the purpose of typedef is to redefine or rename the name of an existing data type.

Syntax - ~~data~~ typedef datatype new name

Eg:- typedef int Number;
typedef float Salary;

The new names that are given by the user for the already existing data types are called user defined data types. In the above example Number and Salary are user defined data types.

→ typedef int Number;
Number n1, n2, n3; //declaration of variables

→ typedef float Salary;
Salary s1, s2, s3;

* Creating a structure using typedef -

There are two ways to creat structures using typedif-

i) After declaration of structure -

```
struct student
{
  ---
  ---
  ---
};
typedef struct student Student;
Student s;
```

ii) At the time of declaring structure -

```
typedef struct student → optional
{
  ---
  ---
} student;
```

*

```
# include <stdio.h>
typedf struct student
{
    char name[20];
    int roll;
    float marks;
}student;
int main()
{
    student s1 = {"SP", 02, 90};
    student s2 = {"PK", 03, 85};
```

Date
Page

printf ("Information of S1");
printf ("\n %.s %.d %.f", S1.name, S1.roll, S1.marks);

printf ("Information of S2");
printf ("\n %.s %.d %.f", S2.name, S2.roll, S2.marks);

* Union -

Union is derived data type contains collection of different data type or dissimilar element.

* The union is much like a structure, so all definition declaration of union variable and accessing member is similar to structure, but instead of keyword 'struct' the keyword 'union' is used.

* Difference b/w union and structure -

i) Structure is defined using the "struct" keyword whereas union is defined using the "union" keyword.

ii) The size of the structure is equal to the total size of all members, whereas the size of union is equal to the largest size member of the union. Due to this reason union is used for saving memory. and this concept is useful when it is not necessary to use all members of union at a time.

iii) In structure all fields data remain available because each field ~~data~~ has its own memory space whereas in union all members do not remain available at the same time because all members of union uses the same location by over-writing.

iv) Structure is suitable for data file handling whereas union does not suitable for data file handling.

* WAP to illustrate union -

```
# include <stdio.h>

union student
{
    char name[15];
    int marks;
    float fee;
};
void main()
{ union student s;
    printf("Enter student name");
        scanf("%s", s.name);
printf("Enter marks "); scanf("%d", &s.marks);
 printf("Enter fee:"); scanf("%f", s.fee);

printf("\n Name : %s", s.name);
printf("\n Marks : %d", s.marks);
printf("\n fee : %.2f", s.fee);
}
```