Structure

C language provides Arrays and Strings to let us handle similar data.
But real word data is usually dissimilar For example, a "book" is a collection of items like title, author, publisher, no. of pages, etc. For dealing with such data C provides a datatype called "Structure".

- Arrays contains multible data / multiple elements but the data should be of similar types.
- Structure contains multiple data or multible elements and data can be of different types.

Structure is a user defined data type which is stored different types of data in a single unit.
Example - Suppose, we want to store the data of the employee, then :-

- emp_code is in integer
- emp_name is in char
- designation is in char
- emp_salary is in float.

structure

* Syntax of a structure :-

```
struct structure_name
{
    statement 1;
    statement 2;
    statement 3;
};
```

* Example of a structure :-

```
struct Employee
{
    int emp_code;
    char emp_name[100];
    char designation[100];
    float salary;
};
```

Now this is our user defined datatype.

The variable declaration of this datatype is like :-

```
struct Employee a;
```

"struct Employee" → name of the datatype

"a" → name of the variable

Now this "a" variable contains all the data which we mentioned inside the structure.

* How to access the elements of the structure ?

```
a.emp_code = 101;
a.emp_name = "aman";
a.designation = "programmer";
a.salary = 50379;
```

NOTE :- the best practice is to create a structure globally in a program.

Program :- create an employee structure & display the same in C.

```
#include <stdio.h>
struct Employee
{
    int Emp_code;
    char Emp_Name[50];
    char Emp_Designation[50];
    float Emp_Salary;
};
int main()
{
    struct Employee emp;
    printf("enter employee code :");
    scanf("%d", &emp.Emp_code);
```

④

```
printf("enter employee name: ");
scanf("%s", &emp.Emp_Name);
printf("enter employee designation");
scanf("%s", &emp.Emp_Designation);
printf("enter employee salary: ");
scanf("%f", &emp.Emp_salary);

printf(" code is %d", emp.Emp_code);
printf("Name is %s", emp.Emp_Name);
printf("Designation is %s", emp.
                    Emp_Designation);
printf("salary is %f", emp.Emp_
                    Salary);
}
```

* Array of Structure

```
#include <stdio.h>
struct Employee
{
    int empcode;
    char empName[50];
    float empSalary;
};
int main()
{
    struct Employee e[10];
    int i, key, flag=0, pos;
    for(i=0; i<10; i++)
    {
        printf("Enter code = ");
        scanf("%d", &e[i].empcode);
        printf("Enter Name = ");
        scanf("%s", &e[i].empName);
```

```
    printf("Enter salary = ");
    scanf("%f", &e[i].empsalary);
  }
  printf("enter code to search = ");
  scanf("%d", &key);
  for(i=0; i<10; i++)
{
    if(e[i].empcode == key)
  {
      flag=1;
      pos=i;
      break;
  }
}
  if(flag==1)
{
    printf("code: %d", e[pos].empcode);
    printf("Name: %s", e[pos].empName);
    printf("salary: %f", e[pos].empsalary);
  }
  else
{
  printf("search item not found");
}
  return 0;
}
```

* Structure as function Argument.

i) A structure can be passed to any function from main function or from any sub function
ii) Structure definition will be available within the function only.
iii) It won't be available to other function unless it is passed to those functions by value or by address (reference).

* Program:- Passing structure as function argument.

```
#include <stdio.h>
struct Employee
{
    int code;
    char name[50];
    float salary;
};
void display(struct Employee aa)
{
    printf("employee code = %d", aa.code);
    printf("employee name = %s", aa.name);
    printf("employee salary = %f", aa.
                                    salary);
}
int main()
{
    struct Employee aa;
    printf("enter code = ");
    scanf("%d", &aa.code);
```

```
printf("enter name = ");
scanf("%s", &aa.name);
printf("enter salary = ");
scanf("%f", &aa.salary);
display(aa);
return 0;
}
```

union

union is also a user defined data type, just like structure, union combines objects of different types & sizes together. The union variable allocates the memory space equal to the space to hold the largest variable of union.
It allows varying types of objects to share the same memory.

Syntax of union :-

```
union employee
{
    int code;
    char name[50];
    float salary;
};
```

* Checking the size of structure variable and union variable.

```
#include<stdio.h>
struct employee
{
  int code;
  char name[50];
  float salary;
}aa;
int main()
{
  printf("Size of aa variable in structure
```

output : Size of aa variable in structure is 60 Bytes.

```
    is %d Bytes ", sizeof(aa));
    return 0;
}
```

```
#include <stdio.h>
union employee
{
    int code;
    char name[50];
    float salary;
};
int main()
{
    printf(" size of aa variable in union
        is %d Bytes ", sizeof(aa));
    return 0;
}
```

output: size of aa variable in union is 52 Bytes.

union is user-defined data type which is used to store different types of data in a single unit. This is much similar to structure, having the following differences :-

a) for creating structure, we use "struct" keyword while for union we use "union" keyword.

b) In structure memory is created for all members while in union, memory is created for the largest member & this memory is shared by rest of the members.

c) In structure, we can retreive any no. of members at a time. In union,

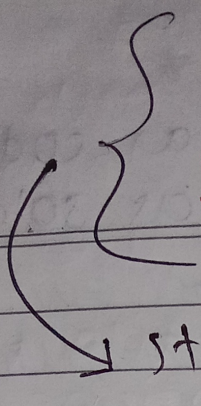
(10) , we can access only one member at a time.

For example :-

```
#include<stdio.h>
struct Employee
{
    int code;
    char name[50];
    float salary;
};
int main()
{
    struct Employee a;
    printf("enter code : ");
    scanf("%d", &a.code);
    printf("enter name : ");
    scanf("%s", &a.name);
    printf("enter salary : ");
    scanf("%f", &a.salary);
    printf("code = %d ", a.code);
    printf("name = %s", a.name);
    printf("salary = %f", a.salary);

    return 0;
}
```

Let suppose, user enters 101 as code, Amit as name and 52143.278 as salary. Then our ouptuput will be :-



structure can access all the members at same time.

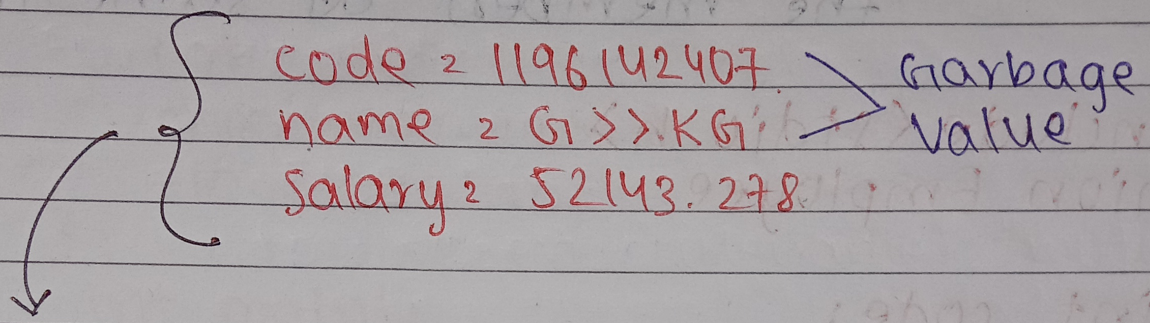
```
#include <stdio.h>
union Employee
{
    int code;
    char name[50];
    float salary;
};
int main()
{
    union Employee a;
    printf("enter code = ");
    scanf("%d", &a.code);
    printf("enter name = ");
    scanf("%d", &a.name);
    printf("enter salary = ");
    scanf("%f", &a.salary);

    printf("code = %d", a.code);
    printf("name = %d", a.name);
    printf("salary = %f", a.salary);

    return 0;
}
```

(12)

Let suppose, user enters 101 as code, Amit as name & 52143.278 as salary.
Then our output will be :-



this will display the last value correctly all the other values will be garbage values because its a union & union can access one member at a time.