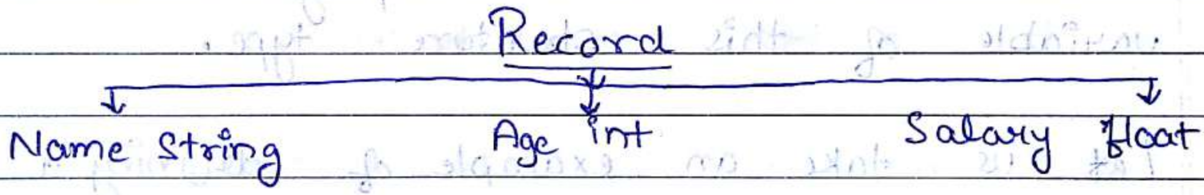
UNIT :→ 4

★ Structure :→

Array is a collection of same type of elements, but In many real life applications, we may need to group different types of logically related data.
For ex:→ If we want to create a record of a person that contains name, age, sallery of that person then can't use array because all the three data elements are of different types.



To store these related things of different datatypes we can use data structure which is capable of storing hetrogenious data.

★ Defining a Structure :→
Create a ~~d~~ template or format that describe the characteristic of its ~~symbol~~ member. All the variables that would be declare of this structure type.

The General Syntax :- The general syntax of structure definiation is :-

```
Struct tagname
{
    datatype member1;
    datatype member 2;
    - - - - - - -
    - - - - - - - -
    datatype member n;
};
```

Here Struct is a keyword which tells the compiler that Structure is being define, member1 member 2 and soon member n, are known as members of the structure. And are declare inside curly braces { }.
Tagname is name of the structure and it is used further in the program to declare variable of this structure type.

Let us take an example of defining a structure template :→

```
Struct Student
{
    char name[20];
    int age;
    float marks;
};
```

Date / /

★ Declaring Structure Variable :→

By defining a structure we have only created a format. The actual use of structures will be when we declare variables based on this format. We can declare structure variable in two ways :→

1. Structure Defination.
2. Using the Structure tag.

① With Structure Defination :-

```
Struct Student {
                  char name[20];
                  int Roll no;
                  float marks;
          } Stru1, Stu2, Stu3;
```

Here, Stu1, Stu2, Stu3 are variable of type struct Student, when we declare a variable while defining the structure template. The tag name is Optional, so we can also declare them as,

```
Struct {
          char name[20];
          int Roll no;
          float marks;
       } Stu1, Stu2, Stu3;
```

If we declare variables in this way, then we will not able to declare other variable of this structure type anywhere in the program.

② Using the Structure tag :-

② Using the Structure tag :-

```
Struct Student {
        char name[20];
        int Roll no;
        float marks;
        };
Struct Student Stu1, Stu2, Stu3;
```

Here, Stu1, Stu2, Stu3 are structure variable that are declare using the structure tag student.

Initialization of Structure Variable :-
The syntax of initializing structure variable is similar to that of arrays.
All the values are given in curly braces { } and the number order and type of these values should be same as in the structure template definition.

```
Struct Student {
        Char name[20];
        int roll no;
        float marks;
        } Stu1 = {"John", 25, 65.5};
Struct Student Stu2 = {"Sinha", 26, 76};
```

Here value of members of Stu1 will be "John" for name, 25 for roll no; 65.5 for marks.
The value of number of Stu2 will be "sinha" for name, 26 for roll no; 76 for marks.

Date / /

☆ Accessing Members of Structure :→
For accessing any member of a structure variable we use dot (.) operator. which is also known as membership operator.
The format or accessing a structure member
struct variable.member

Q19:→ W.A.P to accept the information of your teacher is the following Details (field):→ Nam, department, qualification, desigination ?

```
            #include<stdio.h>
            #include<conio.h>
            #include<string.h>
              void main()
   struct str {
            char name[20];
            char deptartment [20];
            char qualification [20];
            char designation [20];
            };

              void main()
              {
              clrscr();
              str x;

   printf("Enter the Name:");
   scanf("%S", x.name);

   printf("\n Enter the Department:");
   scanf("%s", x. department);
```

```
printf("\n Enter Qualification:");
scanf("%s", x.qualification);

printf("\n Enter the Desigination:");
scanf("%s", x.desigination);

 printf("\n Name is:%s", x.name);
 printf("\n Department is:%s", x.department);
 printf("\n Qualification is:%s", x.qualification);
 printf("\n Desigination is:%s", x.desigination);

 getch();
 }
```

Output:-

```
Enter the Name: Mayank

Enter the Department: BCA

Enter the Qualification: 12th

Enter the Desigination: DBGI

Name is : Mayank
Department is : BCA
Qualification is : 12th
Desigination is : DBGI
```

Date / /

imp ☆ Nested Structure (Structure within Structure):-

The members of a structure can be of any datatype including another structure type, we can include a structure within another structure.

The structure variable can be member of another structure this is called nesting of the structure,

```
Struct tag 1 {
        member 1;
        member 2;
        struct tag 2 {
                member 1;
                member 2;
                ------
                member m;
                } var 1;
        ------
        member n;
        } var 2;
```

For accessing member 1 of inner structure, we will write, var 2 . var1 . member 1.

Here is an example of nested structure :→

```
struct student 1 {
        char name;
        int Roll no.;
            Struct date {
                int date;
                int month;
                int year;
                } birth date;
float marks;
} stu1, stu2;
```

Here we have define a structure date inside the structure (student). This structure date has three members " day, moth, year". Birth date is a variable of type struct data.

We can access the members of inner structure as,

$\boxed{\text{stu1. birthdate. day}}$

Day of birth date of stu1 similarly month similarly year.

v.Imp.
☆ Union :→

Union is a drived datatype like structure and it can also contain members of different datatype. The syntax used of defination of an Union, declaration of union ~~variable~~ and accessing members is similar to that used in structures, but here keyword "union" is used instead of "struct."

The main differente b/w union & structure is in the way memory is allocated for the members.
In the structure, each member has its own memory location whereas members of unions share the same memory location.
When a variable of type Union is declared, compiler allocates sufficient memory

Date / /

to hold the largest member in the union.
' Since, all members share the same memory location hence we can use only one member at a time.
Thus union is used for saving memory.
The syntax of defination of a union is :-

```
union union_name {
        datatype member 1;
        datatype member 2;
        - - - - - -
        - - - - - -
} variable_name;
```

This can also be declare as,

```
union union_name : variable_name;
```

Diff^r b/w Structure and Union :→

Struct student {
int roll;
char name[10];
} stu;
	Union student {
	int roll;
	char name [10];
	} stu;

roll ↳ 2 bytes name ↳ 10 bytes location → 2156 2158 Total 12 bytes.	roll ↳ 2 bytes name ↳ 10 bytes Total → 10 bytes Because it is greater than roll no [2 bytes]

Difference ⇒ Union = 10 bytes
Structure = 12 bytes

Q.→ Write a program to compare the memory allocated for a union and structure variable?

```
#include<stdio.h>
#include<conio.h>
struct stag{
int i;
char c;
float f;
};

union utag{
int i;
char c;
float f;
};

void main()
{
clrscr();
union utag uvar;
struct stag svar;
printf("Size of svar %d\n", sizeof(svar));
printf("Address of svar %u\n", &svar);
printf("Address of member %u, %u, %u\n\n\n", &svar.i,
                    &svar.c, &svar.f);
printf("Size of uvar %d\n", sizeof(uvar));
printf("Address of uvar %u \n", &uvar);
printf("Address of member %u, %u, %u \n", &uvar.i
                    , &uvar.c, &uvar.f);
getch();
}
```