Structure and Union

If we want to create a user defined datatype. So in C Language this can be done using structure & Union.

Predefined Datatypes:-

- int
- Char
- float
- double

Userdefined Datatype:-

- → Employee
- → Student
- → Teacher

```
Char name[20];
int salary;
```

```
char name[10][20];
int salary[10];
```

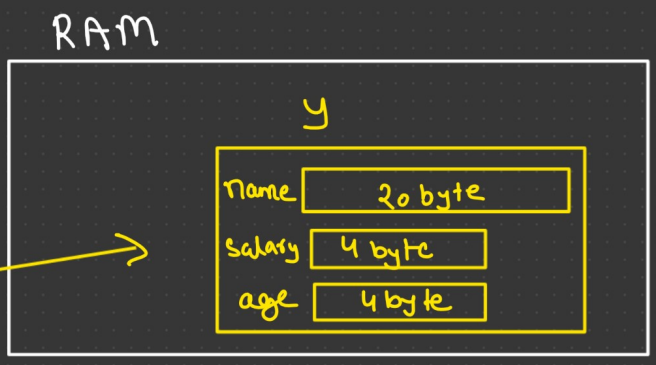
```
struct Employee
{
  char name[20];
  int salary;
  int age;
};
```

→ **Defination of Employee structure**

↳ Blue print of Employee

Declaration of Employee :-

```
int x;
struct Employee y;
```



Total = 28 byte

Intialization of Employee :-

```
struct Employee y = { "prateek", 100, 20 };
```

name salary age

Accessing all the elements/members of Employee :-

```
y.name = "Prateek" , y.salary = 200 , y.age = 21
```

↑

Dot operator

Designated Initialization :-

```
Struct Employee z = { .name = "prateek", .age = 21, .salary = 150};
```

Taking input from user :-

①
```
fflush ( stdin);
fgets ( y.name , 20 , stdin);
```

②
```
scanf (" %d" , &y. salary);
scanf ["%d" , &y.age);
```

Creating array of Employee :-

```
struct Employee e[10];
```
⟶ Declare

```
for( i = 0; i<10 ; i++)
{
    fgets (e[i].name, 20, stdin);
    scanf ( "%d", &e[i].age);
}
```
⟶ Assign

```
for (i=0; i<10 ; i++)
{
    printf(" Name = %s , Age = %d", e.name[i] , e[i].age);
}
```
⟶ Printing.

DMA for struct

```
struct Employee *e = (struct Employee *)malloc( sizeof(struct Employee));
```

```
int * p =(int *) malloc( sizeof(int))
```

int = struct Employee

```
fflush (stdin);
fgets ( e→ name , 20 , stdin);

scanf( "%d" , & e→ age);
```

---

structure without name :-

```
struct
{
    int x;
    int y;
} A, B;
```

```
A.x = 5;
A.y = 10;
```

```
B.x = 10;
B.y = 20;
```

If I want to restrict number of objects created for my datatype.

We can't perform arithmetic operation on struct object.

For ex:- A + B, A - B, A * B ⇒ error

A.x + B.x ⇒ valid

```c
#include<stdio.h>
#include<stdlib.h>
struct Employee
{
    char name[20];
    int age;
};

int main()
{
    struct Employee *e;
    int count,i;

    printf("Enter number of records\n");
    scanf("%d",&count);

    e = (struct Employee *) calloc(count,sizeof(struct Employee));

    for(i = 0 ; i < count ; i++)
    {
        printf("Enter %d Employee name\n",i+1);
        fflush(stdin);
        fgets((e+i)->name ,  20 , stdin);

        printf("Enter its age\n");
        scanf("%d",&(e+i)->age);
    }

    for(i = 0 ; i < count ; i++)
    {
        printf("Name = %s , Age = %d\n",(e+i)->name,(e+i)->age);
    }

    free(e);

    return 0;
}
```

Structure and function

```
int main()
{
  struct Employee A;
  struct Employee *B = (struct Employee *) malloc ( sizeof (struct Employee));

  fun ( A , B );

}
```

```
Void fun ( struct Employee X, struct Employee *Y)
 {
      X.name;
      Y→name;

 }
```

Union

```c
#include<stdio.h>

union demo
{
   int i;
   char ch[2];
};

int main()
{
  union demo a;
  a.i = 32770;
  printf("a.i = %d\n",a.i);
  printf("%d %d %d %d\n",a.ch[0],a.ch[1],a.ch[2],a.ch[3]);
  return 0;
}
// Little Endian Format (R to L)
// 00000000000000001000000000000010  = 32770
```

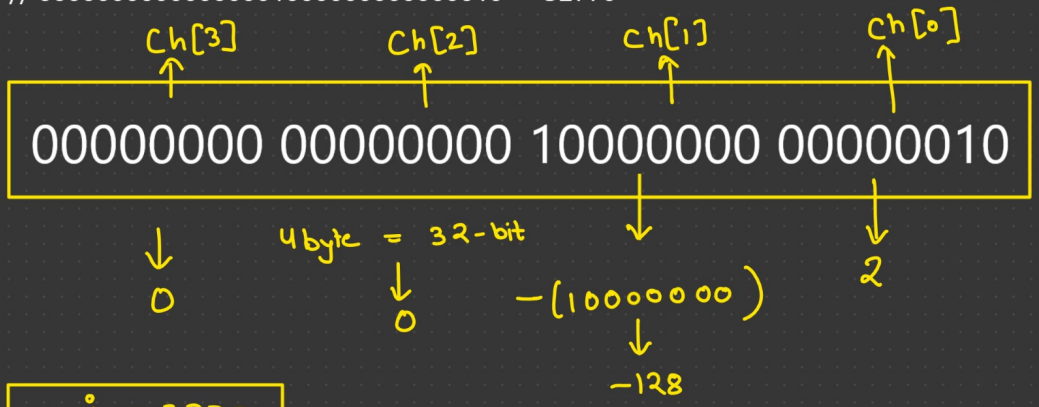
4 byte

2 -128 0 0

```c
union Employee
{
    char name[20];
    int id;
};
```

Structure ⟶ 24 bytes

Union ⟶ 20 bytes.



a.i = 32770

Enum

```
#include<stdio.h>
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};

enum State {Working = 1, Failed = 0, Freezed = 0};

enum day {sunday = 1, monday, tuesday = 5,wednesday, thursday = 10, friday, saturday};



enum state  {working, failed};
//enum result {failed, passed};


enum State currState = 2;
int main()
{
   enum week day;
   day = Wed;
   printf("%d\n",day);

  int i;
  for (i=Mon; i<= Sun; i++)
    printf("%d ", i);

   printf("\n%d, %d, %d", Working, Failed, Freezed);

   printf("\n%d %d %d %d %d %d %d\n", sunday, monday, tuesday,wednesday, thursday, friday, saturday);

   ( currState == Working)? printf("WORKING"): printf("NOT WORKING");
   return 0;
}
```

```
int week[] = { 0, 1, 2, 3, 4, 5, 6};
week[0] = 0
week[1] = 1

if( week[2] == 2)
    printf("Yes");
```

```
enum week {M, T, W, Th, F, S, Sn};
          ↓  ↓  ↓
          0  1  2
```

```
enum week day;

day = w;

if( day == w)
    printf("Yes");
```

```
int day;

day = 2

if(day == 2)
    printf("Yes");
```