

Structure and Union

If we want to create a userdefined 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;  
};
```

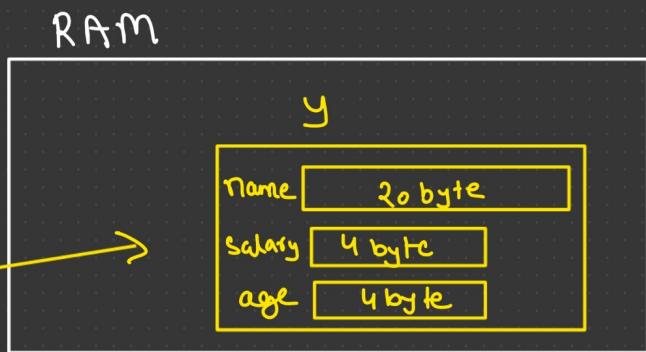
→ Definition of Employee structure

↳ Blueprint of Employee

Declaration of Employee :-

int x;

struct Employee y;



Total = 28 byte

Initialization 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);  
}
```

for (i=0 ; i<10 ; i++) → Printing.

```
{  
    printf (" Name = %s , Age = %d ", e.name[i] , e[i].age );  
}
```

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;
}
```

$$\begin{array}{l} A.x = 5, \\ A.y = 10; \end{array}$$

$$\begin{array}{l} B.x = 10; \\ B.y = 20; \end{array}$$

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

☞ We can't perform arithmetic operation on struct object.

For ex:- $A + B$, $A - B$, $A * B \Rightarrow$ error

$A.x + B.x \Rightarrow$ valid

```
#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

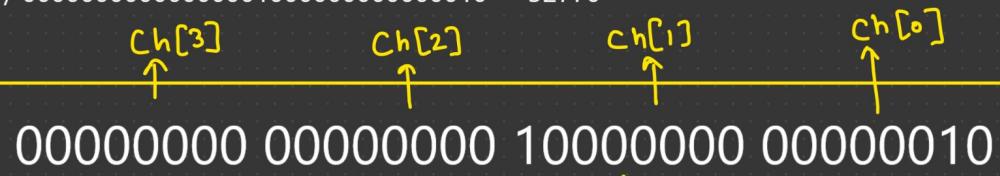
```
#include<stdio.h>
```

```
union demo
{
    int i; → 4 byte
    char ch[2];
};
```

```
int main()
{
    union demo a;
    a.i = 32770;
    printf("a.i = %d\n", a.i); 2 -128 0 0
    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
```

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

Structure → 24 bytes
Union → 20 bytes.



↓ 4 byte = 32-bit
0 ↓ - (10000000)
 ↓
 -128

a.i = 32770

Enum

M, T, W, Th, F, S, Sun

```
#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 week X;

```
enum State currState = 2;
```

```
int main()
```

```
{  
    enum week day;  
    day = Wed;  
    printf("%d\n", day);
```

int day = 2;

1, 2, 5, 6, 10, 11, 12

```
int i; 0 6  
for (i=Mon; i<= Sun; i++)  
    printf("%d ", i);
```

1 0 0

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

```
printf("\n%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;	int day; day = 2
	if(day == w) printf("Yes");	if(day == 2) printf("Yes");

```

int x = 5;
if( y == n)
    =
else
    =

```

while(x != 5)	{
	}

```

#include <stdio.h>
#define LIMIT 5
// Function like macro
#define AREA(l,b) (l * b)
#define MIN(a,b) (((a) < (b)) ? (a) : (b))

// Chain Macro
#define YOUTUBE FOLLOWERS
#define FOLLOWERS 45
int main()
{
    printf("The value of LIMIT is %d\n", LIMIT);

    int area = AREA(2, 4); → 2*4
    printf("Area of rectangle is: %d\n", area);

    printf("Minimum value between %d and %d is %d\n", 5, 10, MIN(5,10));

    printf("PRATEEK JAIN have %d K followers on YOUTUBE", YOUTUBE);
    return 0;
}

```

- ① LIMIT = 5
- ② Area(l,b) = (l*b)
- ③ Min(a,b) = a < b ? a : b
- ④ Followers = 45
- ⑤ Youtuber = Followers

