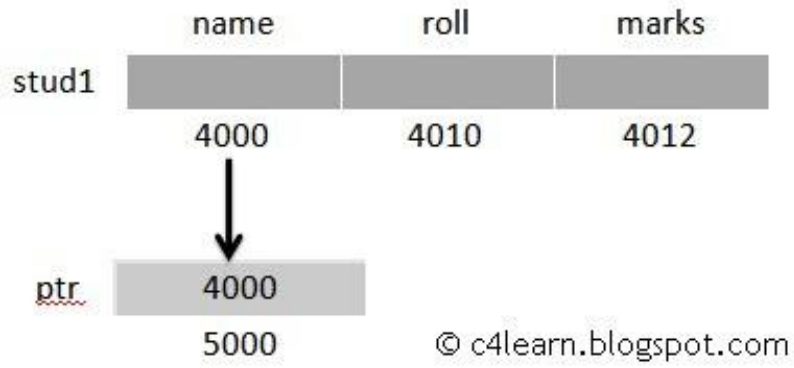
Pointer to Structure:

```
struct student_database
{
    char name[10];
    int roll;
    int marks;
}stud1;
struct student_database *ptr;
ptr = &stud1;
```



How to Access Structure Members : Way 1 : Using Structure Name

```
Accessing Roll Number : stud1.roll
Accessing Name        : stud1.name
```

We can use DOT operator to access the members of the structure.

Live Example 1 :

```
#include <stdio.h>
int main()
{
struct student_database
{
    char name[10];
    int roll;
    int marks;
}stud1;

stud1.roll = 10;
stud1.marks = 90;

printf("Roll Number : %d",stud1.roll);
printf("Marks of Student : %d",stud1.marks);

return 0;
}
```

Way 2 : Using Indirection Operator and Pointer

```
Accessing Roll Number : (*ptr).roll
Accessing Name        : (*ptr).name
```

Pointer variable is declared and pointer to structure. Whenever we declare pointer variable ,address of the structure variable is assigned to the pointer variable.

Live Example 2 :

```c
#include <stdio.h>
int main(int argc, char *argv[])
{

struct student_database {
    char name[10];
    int roll;
    int marks;
}stud1 = {"Pritesh",90,90};

struct student_database *ptr;
ptr = &stud1;

printf("Roll Number : %d",(*ptr).roll);
printf("Marks of Student : %d",(*ptr).marks);

return 0;
}
```

Way 3 : Using Membership Operator

```
Accessing Roll Number : ptr->roll;
Accessing Name        : ptr->name;
```

Live Example 3 :

```c
// assuming there is a structure variable already created same as last eg.

struct student_database *ptr;
ptr = &stud1;

printf("Roll Number : %d",(ptr)->roll);
printf("Marks of Student : %d",(ptr)->marks);

return 0;
}
```

Whenever we access the member of the structure using the pointer we use arrow operator to access the member of structure.