Structures: overview

- Definition & structure variable declaration

```
struct student
  {  int rollno;
     int age;
     char name[20];
     }s1, s2, s3;
```

- Initialization

```
int main( ){
struct
 { int rollno;
      int age;
      }stud={20, 21};
      …
      …
      return 0;
}
```

- Giving values to members

Using dot operator '.'

```
s1. rollno = 25;
cin>>s1.name;
```

'.' operator acts as Link between member and a Structure variable.

- Assign & compare members

s1 = s2 ; assignment (allowed)

---

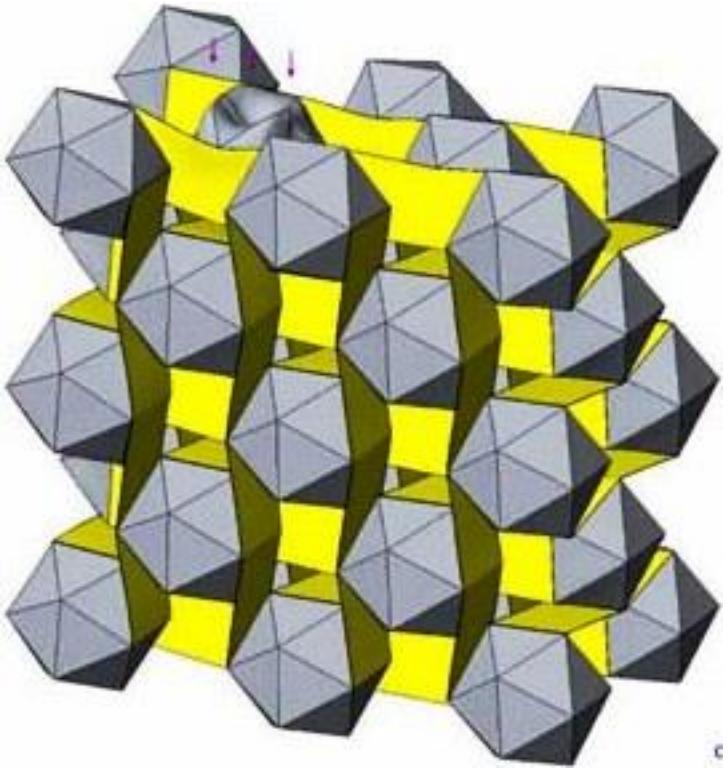
s1 == s2 comparison (not allowed)

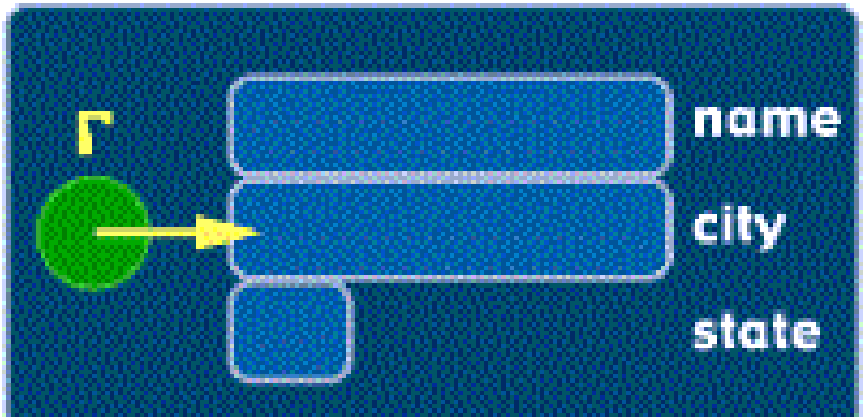
s1!=s2 comparison (not allowed)

---

s1.rollno == s2.rollno; (allowed)

s1.rollno!=s2.rollno; (allowed)





Array of Structures & Pointers to Structures

Objectives

- To learn and appreciate the following concept
  - Array of structures
  - Pointers and Structures

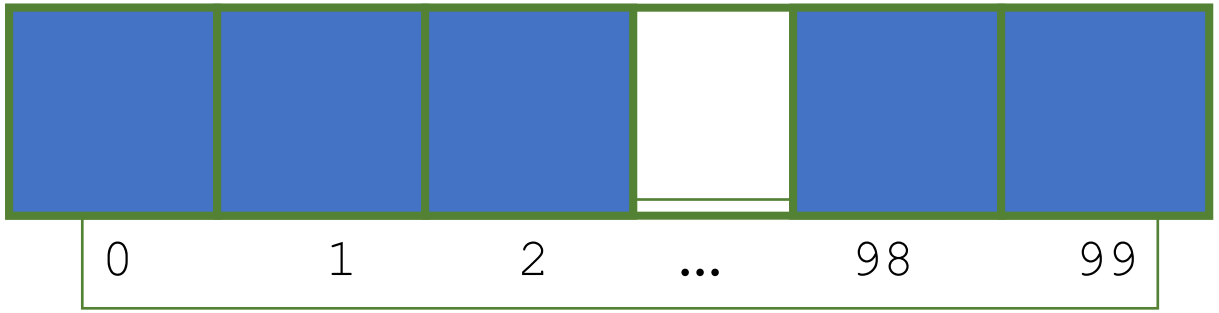
Session outcome

- At the end of session one will be able to

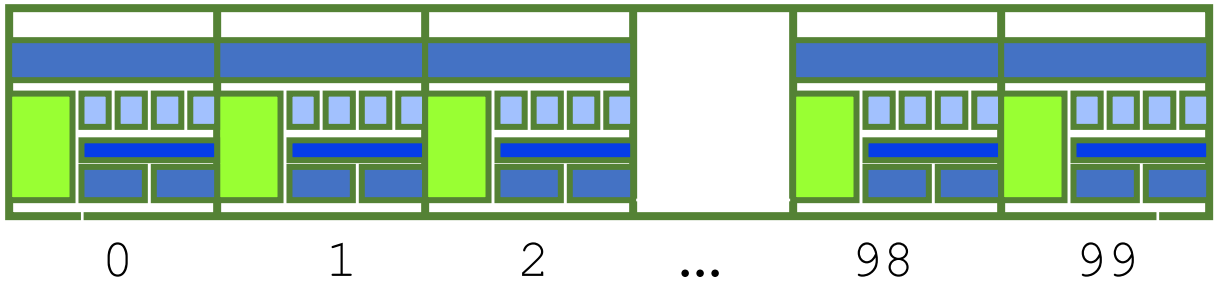
  - Understand the overall ideology of array of structures
  - Write programs using array of structures
  - Understand the concept of pointers to structures
  - Write programs on pointers to structures.

Arrays of structures

- An ordinary array: One type of data



- An array of structs: Multiple types of data in each array element.



Array of structures

We can define single or multidimensional arrays as structure variables.

```
struct marks
  {
      int  subject1;
      int  subject2;
      int  subject3;
  } ;
marks student[80];
```

- Defines an array called student, that consists of 80 elements.
- Each element is defined to be the type marks.

Array of structures – Initialization

```
struct marks {
  int  subject1;
  int  subject2;
  int  subject3;
} ;

main(){
  marks student[]={
                    {45,47,49},
                    {43,44,45},
                    {46,42,43}
                  };
```

	Memory
student[0].subject1	**45**
student[0].subject2	**47**
student[0].subject3	**49**
student[1].subject1	**43**
student[1].subject2	**44**
student[1].subject3	**45**
student[2].subject1	**46**
student[2].subject2	**42**
student[2].subject3	**43**

Array of Structure: Example

```
struct Book {           //Structure Definition
        char title[20];
        char author[15];
        int pages;
        float price;
};
int main( ){
    struct Book b[10];
    printf("Input values");
    for (int i=0;i<3;i++)
    scanf("%s %s %d %f", b[i].title, b[i].author, &b[i].pages, &b[i].price);
    for (int j=0;j<3;j++)
    printf("%s\t %s\t %d\t %f\n", b[j].title, b[j].author, b[j].pages, b[j].price);
 return 0;
}
```

Arrays within Structures

We can define single or multidimensional arrays inside a structure.

```
struct marks
{     int  rollno;
      float  subject[3];
} student[2] ;
```

The member **subject** contains 3 elements; **subject[0], subject[1] & subject[2].**

student[1].subject[2];

- **Refers to the marks obtained in the third subject by the second student.**

Arrays within structures : example

```
#include<stdio.h>
 int main(){
  struct marks student[3] ={{0,45,47,49},
                            {0,43,44,45},
                            {0,46,42,43}};
  int i, j ;
  //students total
  for(i=0;i<=2;i++)  {
    for(j=0;j<=2;j++)
       student[i].total+=student[i].sub[j]; }
  printf("Grand Total of each student:");

for(i=0;i<=2;i++)
   printf("\nTotal of student[%d]= %d", i, student[i].total);
 return 0;
}
```

```
//Structure Definition
struct marks{
   int total;
   int sub[3];
};
```

Structures within Structures

Structure within structure means nesting of structures.

for instance see the following structure defined to store information about students

```
struct student{
   int rollno;
   char name[15];
   struct { // marks for 3 subjects under structure marks
        int sub1;
        int sub2;
        int sub3;
    }marks;
}fs[3]; //3 students
```

Structures within Structures

```
//Structure Definition
 struct student{
    int rollno;
    char name[15];
    struct m marks;
 }fs[3];
```

```
//Structure Definition
struct m{
   int sub1;
   int sub2;
   int sub3;
    };
```

Tag name is used to define inner structure **marks**

The members contained in the inner structure namely **sub1, sub2** and **sub3** can be referred to as:

```
fs[i].marks.sub1;
fs[i].marks.sub2;
fs[i].marks.sub3;
```

Structures and functions

```
void read(struct book x[]); // prototype
int main() {
int i;
struct book b1[2];
printf("\n Enter IBN, Author name & Price \n");
  read(b1); // function call

printf("\nThe book details entered are:\n");
for(i=0;i<2;i++){
  printf("\n Book %d", i+1);
  printf("\nIBN: \t\t%d", b1[i].ibn);
  printf("\nAuthor: \t%s", b1[i].author);
  printf("\nPrice: \t\t%f", b1[i].price);
 }
return 0;
}
```

```
//Structure Definition
struct book
{
  int ibn;
  char author[15];
  float price;
};
```

```
//function definition
void read(struct book a[])
{
 int i;
 for(i=0;i<2;i++){
   printf("\nBook %d\n", i+1);
   scanf("%d", &a[i].ibn);
   scanf("%s", a[i].author);
   scanf("%f", &a[i].price);
  }
}
```

Structures -Problems

Write programs to

1. Create a student record with name, rollno, marks of 3 subjects (m1, m2, m3). Compute the average of marks for 3 students and display the names of the students in ascending order of their average marks.
2. Create an employee record with emp-no, name, age, date-of-joining (year), and salary. If there is 20% hike on salary per annum, compute the retirement year of each employee and the salary at that time. [standard age of retirement is 55]

Structures – Solution for Q1

```
int main()
{
 struct student temp, fs[3] =
            {{1,"manish",45,47,49},
             {2,"ankur",43,44,45},
                        {3,"swati",46,42,43}};
  int i, n=3, total[3]={0}, avg[3]={0},tot=0;

  for(i=0; i< n; i++)   {
    total[i]=fs[i].marks.sub1+fs[i].marks.sub2+
    fs[i].marks.sub3; //students total

   avg[i] = total[i]/3;
 }
//display
printf("Total & Average of each student.\n“);
for(i=0;i<n;i++){
 printf("\nTotal of %s = %d & avg = %d", fs[i].name, total[i], avg[i]);
}
```

```
struct student{
int rollno;
char name[15];
    struct {
            int sub1;
            int sub2;
            int sub3;
            }marks;
};
```

Structures – Solution for Q1

```
// sorting
for(i=0;i<n;i++)
   for(int j=i+1;j<n;j++)
   if(avg[i] > avg[j])
    {
      temp=fs[i]; //Swapping
      fs[i]=fs[j];
      fs[j]=temp;
     }
 for(i=0;i<n;i++) //Sorted list w.r.to average marks
  printf("\n%s\n",fs[i].name);
return 0;
} //end of main
```

Pointers and structures

Consider the following structure

```
struct inventory {
char name[30];
int number;
float price;
} product[2],*ptr;
```

This statement declares product as an array of 2 elements, each of the type struct inventory.

ptr=product; assigns the address of the **zeroth** element of **product** to **ptr** or **ptr** points to **product[0];**

Pointers and Structures

Its members are accessed using the following notation

ptr →name
ptr →number
ptr →price

The symbol → is called arrow operator (also known as member selection operator)

When **ptr is incremented by one**, it points to the next record. i.e. **product[1]**

The member price can also be accessed using

(*ptr).price

Parentheses is required because **'.'** has higher precedence than the operator *

Pointers and Structures- example

```
struct invent{
 char name[30];
 int number;
 float price;
};
```

```
int main() {
  struct invent prod[3], *ptr;
  printf("Enter  3 (0, 1 and 2 )sets of Name,
  Number and Price");

  for(ptr = prod; ptr < prod+3; ptr++)
     scanf("%s %d %f",ptr ->name, &ptr ->number, &ptr ->price);

  ptr=prod;

  while(ptr < prod+3)  {
    printf("%s  %d  %f\n", ptr ->name,
              ptr ->number, ptr ->price);     ptr++;
  }
  return 0;
}
```

```
Enter  3 (0, 1 and 2 )sets of Name , Number and Price
c_Book
100
250
C++Book
200
350
Java
150
400
c_Book  100  250
C++Book  200  350
Java  150  400

Process returned 0 (0x0)   execution time : 33.424 s
Press any key to continue.
```

Summary

- Array of Structures
- Arrays within Structures
- Structures within Structures
- Structures and Functions
- Pointers and Structures