27/3/2016

DATE: PAGE:

Chaithra H.E

Module 5 :-

~~Structures~~ (~~Refer~~ PPT)

CHAITHRA. H.E.
B.E, M. Tech.,

~~pointers~~ :-

Structures :- (Part 1)

- Introduction to structure
- Declaration & Initialization of structures
- Accessing members of structure
- Passing structure to function
- Array of structure

- Introduction to Structures

structure is a collection of variables of different types under a single name.

For eg., : You want to store some information about a person: his/her name, salary. You can easily create different variables name, salary to store information separately.

Instead of using different variables we can create structure person which contains all the details about the person.

Keyword struct is used for creating a structure.

Syntax

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memberN;
};
```

CHAITHRA. H.E.
B.E, M. Tech

Example

```
struct person
{
    char name[50];
    int id;
    float salary;
};
```

This above declaration creates the derived datatype struct person.

P.T.O.

DATE: PAGE:

Structure variable declaration

Memories will be allocated to members only after structure variable.

There are 2 methods to declare structure variable. ↓

structure variable declarati

1. Structure variable declaration inside the main()

For the above structure of a person, variable can be declared as:

```
struct person
{
  char name[50];
  int id;
  float salary;
};
void main()
{
  struct person person1;
}
```


In this example, the structure variable person1 is declared inside the main() function.

2. Structure variable declaration in structure itself.

For the above structure of a person, variable can be declared as:

```
struct person
{
    char name[50];
    int id;
    float salary;
} person1;
```

CHAITHRA. H.E.
B.E, M. Tech

Initialization of structure variables

1. Initializing structure variable in the structure (single structure variable)

Example:

```
struct person
{
    char name[50];
    int id;
    float salary;
} person1 = {"ABC", 10, 2000};
```

Here all the fields or members of the single structure variable initialized.

DATE: PAGE:

2. Initializing structure variable in the structure (Multiple structure variables)

Eg:- struct person
{
char name[50];
int id;
float salary;
} person1 = {"Roshini", 10, 30000}, person2 = {"Chaithra", 20, 35000};

Here, all the fields are members of the multiple (2) structure variables are initialized.

3) Initializing single member of structure

Eg:- struct person
{
char name[50];
int id;
float salary;
} person1 = {"Roshini"};

Though there are 3 members of structure, only one is initialized, then remaining 2 members are initialized with 0 & 0.000000.

Data type & their initial values

Data type	default value
integer	0
float	0.000000
char	Null

4) Initializing structure inside main()

Eg:- struct person
{
char name[50];
int id;
float salary;
} person1;

void main()
{
struct person person1 = {"Roshini", 10, 30000};
}

Accessing members of structure

Array elements are accessed using the subscript variable, similarly structure members are accessed using dot [.] operator.

[.] is called as "Structure Member Operator".

use this operator in b/w "Structure variable name" & "member name".

Eg:- #include <stdio.h>
struct student
{
int rollno;
char name[20];
int marks;

} student1 = { 4, "RoshiniRaj", 95 };

void main()
{
clrscr();
printf("Rollno: %d \n", student1.rollno);
printf("Student name: %s \n", student1.name);
printf("Marks: %d", student1.marks);
getch();
}

Passing structure to function

A structure variable can be passed to the function as an argument as a normal variable.

If structure is passed by value, changes made to structure variable inside the function definition does not reflect in the originally passed structure variable.

Write a program to create a structure student, containing name & rollno & display the information.

```
#include <stdio.h>
#include <conio.h>
struct student
{
  char name[50];
  int roll
```

```
} stud;

void output(struct student stud);

void main()
{
    clrscr();
    printf("Enter student name\n");
    scanf("%s", &stud.name);
    printf("Enter roll no\n");
    scanf("%d", &stud.roll);
    output(stud); // function call
    getch();
}

void output(struct student stud)
{
    printf("Student name is %s\n", stud.name);
    printf("Roll no is %d", stud.roll);
}
```


Array of structure

Structure is used to store the information of one particular object, but if we want to store such 100 objects then array of structure is used.

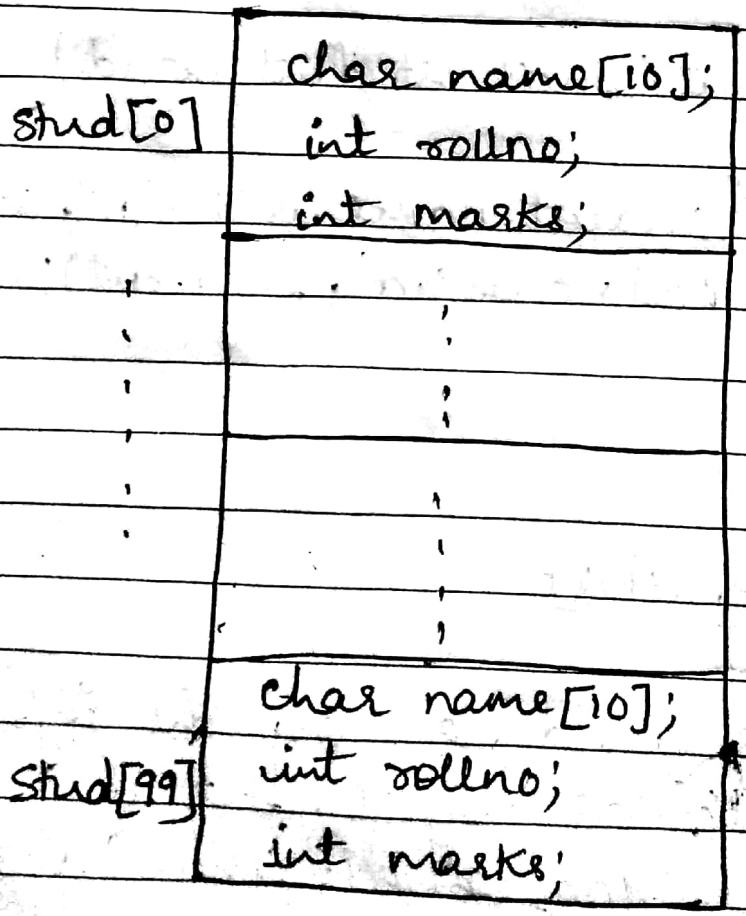
Whenever the same structure has to be applied to a group of people, item we

will use an array of structure.

Example:-

```
Struct student
{
  char name[10];
  int rollno;
  int marks;
} stud[100];
```


The array of structure will be shown as follows:



Lab program 9

WAp to create a structure called Employee to maintain a record of details using an array of structure with fields (Emp_name, Emp_id, Emp_age, Emp_sal). Assume appropriate data type for each field. Print the Employee details in tabular format.

```
#include<stdio.h>
#include<conio.h>
struct employee
{
 char name[20];
 int id, age;
 float sal;
} e[50];

void main()
{
 int n, i;
 float salary;
 clrscr();
 printf("\n Enter the number of employees\n");
 scanf("%d", &n);
 for(i=0; i<n; i++)
 {
 printf("Enter the details of employee %d\n",
                                    i+1);
```

```
printf("Enter name:");
gets(e[i].name);
printf("Enter id:");
scanf("%d", &e[i].id);
printf("Entere age:");
scanf("%d", &e[i].age);
printf("Entere salary:");
scanf("%f", &salary);
e[i].sal = salary;
}
printf("\n ----------\n");
printf("Empname\t Empid\t Empage\t
        Empsal\n");
for(i=0; i<n; i++)
{
printf("\n%s\t%d\t%d\t%f\n", e[i].name,
 e[i].id, e[i].age, e[i].sal);
}
getch();
}
```

23/10/2017

Prof. Chaithra H.E.

DATE: PAGE:

pointers (part 2)

Introduction to pointers
Declaration of pointers
Initialization of pointers
pointers arithmetic
pointers to an array
Function using pointers

CHAITHRA. H.E.
B.E, M. Tech

* Introduction to pointer

pointer is a variable which contains the address of another variable.

It contains only the memory location of variable, but not the value.

pointer can → points to variable of basic types

→ points to array

→ points to function

→ points to structure

* Declaration of pointer

General syntax

type *var-name;

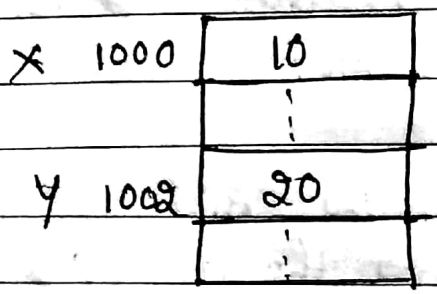
Eg:- int *px; // pointer to int
double *px; // pointer to double

Accessing value by pointer

Consider the following example,

```
int x, y;
x = 10;
y = 20;
```

The memory locations are as follows:



If we want to access value in x via pointer, then we need pointer variable. So assume px as the pointer variable to x. Similarly py for y.

So declare int *px;
int *py;

Now, px = &x;
py = &y;

To fetch the value present in the address we use dereference operator (*).

Dereference operator points to the content of variable.

write a program to print address of variable & its value.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int x=10;
    int *px;
    clrscr();
    px=&x;
    printf("Address=%d\n", px);
    printf("value=%d\n", *px);
    getch();
}
```


* Initialization of pointers

pointers initialization is the process of assigning address of another variable to pointer variable.

There are 3 ways to initialize pointer

1) Declare variable, pointer, assign address of variable to pointer variable separately

Eg:- int a;
int *pa;
pa = &a;

2) Declare variable then declare & initialize pointer

Eg:- int a;
int *pa = &a;

3) Declare variable, pointer & initialize in a single line

int a, *pa = &a;

* WAP to find size of pointer

```
void main() {
    int *iptr;
    char *cptr;
    int s1; int s2;

    s1 = sizeof(iptr);
    s2 = sizeof(cptr);

    printf("Size of iptr = %d \n", iptr);
    printf("size of cptr = %d", cptr);
}
```

o/p:- 2
2

No matter, whether pointer to an char,

int, float, double, the size of pointer will always be 2 bytes. i.e., to store pointer variable we require 2 bytes.

* pointer Arithmetic

* The integer value can be added or subtracted to pointer variable.

Eg:- Addition of integer value to pointer

```
void main()
{
int a=10;
int *pa;
pa=&a;
printf("%d\n", pa); //Actual address
pa=&a+1; /*2 bytes is added to
                actual address */
printf("%d", pa);
getch();
}
```


Eg:- subtraction of integer value from subtraction

```
Void main()
{
int a=10;
int *pa;
pa=&a-1;
```

```
printf("%d", pa);
getch();
}
```

Note:- ptr subtraction or addition will be done to the address to which the pointer is pointing based on the data type of variable.

* 1 ptr variable can be subtracted from another pointer variable, if both are pointing to elements of same array.

eg:-
```
int x[4] = {10,20,30,40};
int *p1, *p2, *p3;
p1 = &x[1];
p2 = &x[2];
p3 = p2 - p1;
```

Suppose &x[1] = 1002 // address
&x[2] = 1004

then, p3 = p2 - p1
p3 = 1004 - 1002
p3 = 2 //

i.e., p3 points to address 2.

x[0]	x[1]	x[2]	x[3]
10	20	30	40
1000	1002	1004	1006

Address

2 bytes will be allocated for each array element.

Pointers increment / decrement

We can perform increment / decrement operation on the pointer.

Eg:-
```
void main()
{
    float a=10;
    float *p;
    p=&a;
    printf("%d\n", p);
    p++;
    printf("%d\n", p);
    p--;
    printf("%d", p);
}
```


Suppose &a = 1000

So p will be 1000

p++ = 1004 (float data type 4 bytes)

p-- = 1000 (decrements address by 4 bytes)

Function using pointer (pass by reference)

Note:- Refer Functions notes for theory

① Eg: Adding 2 nos using call by reference

```
#include<stdio.h>
#include<conio.h>
void
int add(int *a, int *b);
void main()
{
    int a, b, c;
    a=10;
    b=20;
    c = add(&a, &b); // Funct^n call
    printf("%d", c);
}
int add(int *a, int *b)
{
    int c;
    c = *a + *b;
    return c;
}
```

② Eg:- Swapping of 2 nos using call by reference

```
#include<stdio.h>
#include<conio.h>
void swap(int *a, int *b);
void main()
{
    int a,b;
    a=10;
    b=20;
    swap(&a,&b); // Functn call
}
void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
    printf("After swapping a=%d\t
        b=%d", a, b);
}
```

DATE: PAGE:

Pointers & arrays

Pointers can be used with array for efficient programming. In this topic we will come to know how the individual elements of an array can be referenced.

Eg:- int a[5];

where, &a[0] is address of 1st element
&a[1] is address of 2nd element
For ith element,
&a[i-1] is address of ith element

similarly address of (i+1)st element can be written as &a[i] or (a+i).

The value which is present inside the address (&a[i]) can be fetched by a[i] or *(a+i)

Both are same


```
int a[5];
int *p;
p=a; // p=&a[0];
```

the pointer automatically points to first array element's address. So need not to specify address of array while assigning address of array (a) to pointer

10) Write a C program using pointers to compute the sum, mean & standard deviation of all elements stored in an array of n real numbers.

program :-

```
#include<stdio.h>
#include<math.h>
void main()
{
  float a[10], *ptr, mean, std, sum=0,
  sumstd=0;
  int n,i;
 printf("Enter the number of elements\n");
 scanf("%d",&n);
 printf("Enter array elements\n");
 for(i=0;i<n;i++)
  scanf("%f", &a[i]);
 ptr=a;
 for(i=0; i<n; i++)
 {
     sum = sum + *ptr;
     ptr++;
 }
 mean= sum/n;
 ptr=a;
```

```
for (i=0; i<n; i++)
{
    sumstd = sumstd + pow((*ptr - mean),2);
    ptr++;
}

std= sqrt(sumstd/n);
printf("sum =%.3f\n", sum);
printf(" mean =%.3f\n", mean);
printf(" std deviation =%.3f\n", std);
getch();
}
```

$$\text{Sum} = x_1 + x_2 + x_3 + \cdots + x_n$$

$$\text{Mean} = \mu = (x_1 + x_2 + \cdots + x_n) / N$$

$$= \text{Sum} / N$$

Standard deviation

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}$$