

Chapter 9 - Structures

Arrays and strings → Similar data (int, float, char).

Structures can hold → Dissimilar data

A C. structure can be created as follows

struct Employee

```
int code; // This code declares a new user  
float salary; defined data type!  
char name[10];
```

}; // Semicolon is important

We can use this user defined data type as follows:

```
struct Employee e1; // Creating a structure variable  
strcpy(e1.name, "Harry"); // int is used  
e1.code = 100; // int is used  
e1.salary = 21.22; // float is used
```

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

Quick Quiz :- write a programme to store the details of 3 employees from user defined data use the structure declared above

Program to store staff details in a structure

STRUCTURES

Structures help us to store variables & variables have memory
of their relationships or links and relationships
are helpful in program design and maintenance.

Advantages of structures

Now we can easily store data in three types
 (a) Structured data
 (b) Unstructured data
 (c) Semi-structured data
 between the variables there is no link.

What is a struct data type? What is it used for?

Why use structures?

In the structures, we can create the data of different types in the employee structure separately but when the number of properties in a structure increases it becomes difficult for us to create data variables without structures in a midshell structure or by defining them in a separate variable.

(a) Structures keep the data organized

(b) Structures make data management easier for building the programs.

Array of Structures

Just like an array of floats and an array of characters

we can create an array of structures

Struct Employee facebook [100];

facebook [0]. code = 100;

facebook [1]. code = 101; ... (100 to 109)

Initialising Structures

Structures can also be initialised as follows:

Struct Employee Harry = {100, 71.22, "Harry"};

Struct Employee Shubh {0};

Structures in Memory

Structures are stored

in contiguous memory locations. For the structure 'Employee' type struct employee, memory layout looks like this:

100	71.22	"Harry"
-----	-------	---------

Address of Harry → 78810, 78811, 78812, 78813, 78814, 78815, 78816, 78817, 78818

In an array of structures, these Employee instances are stored adjacent to each other.

pointer to Structures

A pointer to Structures can be created as follows:

```
struct Employee { *ptr; };
```

`ptr = &e1;`
`// now we can print structure elements using:`

```
printf("%d", (*ptr).code);
```

Arrow Operator

Instead of writing `(*ptr).code` we can use **arrow operator** to access structure properties as follows

```
(*ptr).code
```

OR

```
ptr->code
```

Passing Structure to a Function

A structure can be passed to a function just like any other data type

```
void show(struct Employee e); // function prototype
```

Quick Quiz: Complete the following function.

Complete this show function to display the content of Employee

object ref. to

memory of object and its contents.

2)

just a point

what are also known as alias

which mean different memory location have the

(2) (a) (i) (a)

the following - P. refresher

and sometimes there can be more than one

alias which are

multiple times

19 bits

19 bits

multiple times

ESD or pattern bytes

it will be stored in memory

Typedef Keyword

we can use the 'typedef' keyword
to create an alias name for data types in C.

'typedef' is more commonly used with structures

it is basically structure type for float, int,

Struct Complex - just add Complex;

it is basically structure type for float, int,

float real;

float img;

3; float real;

float img;

3;

float real;

float img;

3;

Example Usage

Using the `typedef alias`, you can declare complex number variable more succinctly.

(ComplexNo c1, c2);

Chapter 9 - Practice Set

(Q1) Create a two-dimensional structures in C.

#include <stdio.h>

struct vector {

int i;

int j;

};

int main() {

struct vector v = {1, 2};

printf("The value of vector is %d + %d i",

v.i, v.j);

return 0;

(Q2) write a function 'SumVector' which returns the sum of two vectors passed to it. The vectors must be two-dimensional.

#include <stdio.h>

typedef struct vector {

int i;

int j;

}; v;

V SumVector (V v1, V v2)

V v3 = {v1.i + v2.i, v1.j + v2.j};

return v3;

g

```

int main()
{
    // code to initialize v1 and v2
    V v1 = {1, 2};
    V v2 = {5, 6};

    V v3 = SumVector(v1, v2);
    printf("The value of vector v3 is (%d, %d)\n", v3.i, v3.j);

    return 0;
}

```

Q3 Twenty integers are to be stored in memory. What will you prefer - Array or Structure?

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

```
typedef struct Employee
{
    int Salary;
    float Score;
}
```

```
int main()
```

```
Employee e1;
```

```
Employee *ptr = &e1;
```

```
(*ptr).Salary = 56; here we
```

```
(*ptr).Score = 45.3; here we
```

```
// ptr->Salary = 56;
```

```
// ptr->Score = 45.3; here we
```

```
printf("The value of Salary is %d and the value  
of Score is %.2f\n", ptr->Salary,  
ptr->Score);
```

```
return 0;
}
```

Q4) write a program to illustrate the use of arrow operator in C.

3) `#include <stdio.h>`

`typedef struct { int real; int imaginary; } Complex;`

`int main () {`

`Complex c = {1,2};`

`printf("The value of complex number is %d + %d i", c.real, c.imaginary);`

`return 0;`

Q5) Write a program with a structure representing a complex number

3) `#include <stdio.h>`

`typedef struct c {`

`int real; int imaginary;`

`} Complex;`

`void display (Complex c)`

`printf("The value of complex number is %d + %d i", c.real, c.imaginary);`

`int main () {`

Complex Carr[5];
 for (int i=0; i< 5; i++)

printf ("Enter real part \n");
 Scanf ("%d", &Carr[i].real);
 printf ("Enter imaginary part \n");
 Scanf ("%d", &Carr[i].imaginary);
 display (Carr[i]);

return 0;

Q6) Create a Structure representing a bank account of a customer what fields did you use and why?

#include <stdio.h>

Struct bankacc {

int accNo;

char name [84];

char ifsc [12];

float balance;

};

int main()

return 0;

Q7) Write a structure capable of storing dates write a function to compare those dates

```
#include <stdio.h>
typedef struct Data
{
    int mmj;
    int dd;
    int yyyy;
} DT;

int compare(DT d1, DT d2)
{
    if ((d1.yyyy == d2.yyyy) && (d1.mm == d2.mm) && (d1.dd == d2.dd))
        return 0;
    if (d1.yyyy > d2.yyyy)
        return 1;
    else if (d1.yyyy < d2.yyyy)
        return -1;
    else if (d1.mm > d2.mm)
        return 1;
    else if (d1.mm < d2.mm)
        return -1;
```

else if ($d_1.dd > d_2.dd$)
{ return 1;
}

RANKA

DATE

PAGE

else if ($d_1.dd < d_2.dd$)

{ return -1;
}

3) statement in compare function is

void int main() {
 float d1 = 12.4, d2 = 12.5;
 cout << friend float compare(dt, dt);

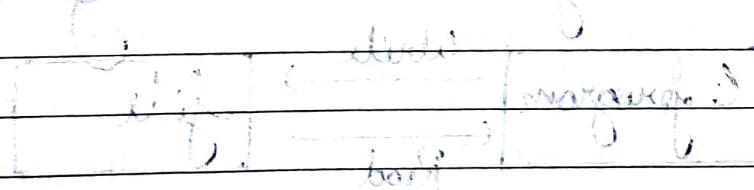
DT d1 = {12, 4, 2154};

DT d2 = {12, 8, 2154};

cout << print("%d", compare(d1, d2)); } // If A
Statement 0;

Output: 1 (d1 is greater than d2)

Q. What is the output of the following program?



What will be the output of the following program?

int main() {
 float d1 = 12.4, d2 = 12.5;
 cout << friend float compare(dt, dt);

DT d1 = {12, 4, 2154};

DT d2 = {12, 8, 2154};

cout << print("%d", compare(d1, d2)); } // If A

Output: 1 (d1 is greater than d2)