JEPPIAAR INSTITUTE OF TECHNOLOGY

"Self-Belief | Self Discipline | Self Respect"

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

LECTURE NOTES

CS8251 – C PROGRAMMING

(Regulation 2017)

Year/Semester: I/II CSE

2020 – 2021

Prepared by

Mr.H.Shine

Assistant Professor/CSE

e1.doj.dd

e1.doj.mm

e1.doj.yyyy

8M

(x) **Array of Structures:-**

✓ There can be array of structures in C-programming to store many information of different data types.

✓ The array structures is also known as collection of structures.

Example: [Stores 5 students information and prints it with array of structure].

```
#include<stdio.h>
#include<string.h>
struct student
  {
  int rollno;
  char name[10];
  };
  int main()
   {
 int i;
 struct student st[5];
 printf("Enter Records of 5 students\n");
 for(i=0;i<5;i++)
  {
  printf("\n Enter roll no:");
   scanf("%d",&st[i].rollno);
   printf("\n Enter name:");
   scanf("%s",&st[i].name);
  }
printf("\n student Information List:");
```

```
for(i=0;i<5;i++)
  {
  printf("\n Rollno: %d, Name: %s",
          st[i].rollno, st[i].name);
  }
return 0;
}
```

Output:-

```
Enter Records of 5 students

Enter rollno: 1
Enter Name: Soniya
Enter rollno: 2
Enter name: Yohesh
Enter rollno: 3
Enter name: John
Enter rollno: 4
Enter name: Mercy
Enter rollno: 5
Enter name: david
Student Information List

Rollno: 1  Name: Soniya
Rollno: 2  Name: yohesh
Rollno: 3  Name: John
Rollno: 4  Name: Mercy
Rollno: 5  Name: david
```

8M

Dynamic memory Allocation :-

✓ The concept of dynamic memory allocation in c-language enables the C-programmer to allocate memory at runtime.

✓ Dynamic memory allocation in C-language is possible by 4 functions of stdlib.h header files.

1. malloc ()
2. calloc ()
3. realloc ()
4. free ()

methods	description
malloc ()	✓ Allocates single block of requested memory.
calloc ()	✓ Allocates multiple block of requested memory.
free ()	✓ Frees the dynamically allocated memory.
realloc ()	✓ Reallocates the memory occupied by malloc () or calloc () functions.

16M

A Self Referenial Structure:-

→ Self Referential structure is the data structure, which contains a pointer, to itself as known as Self Referential.

→ It is used in the many of the data structures like in Linked List, Trees, Graphs, Heap etc.

→ It is a structure which contains pointer variable and points to itself is known as self Referential structure.

→ members are accessed by (→) arrow operators.

Applications:

→ self Referential structure are very useful in creation of other Complex data structure like.

1) Linked List
2) Stack
3) Queues

4) Trees
5) Graphs etc.

2M Linked List:

It is a collection of nodes which is ~~no stored~~ not necessary, to store in adjacent memory location.

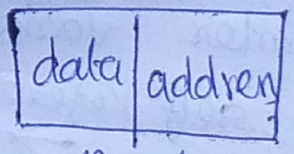
Types of linked list:-

1) Single linked list
2) Double linked list
3) Circular linked list.

Single linked list (SLL):-

In single linked list node has 2 fields. These are

1) data field
2) address field



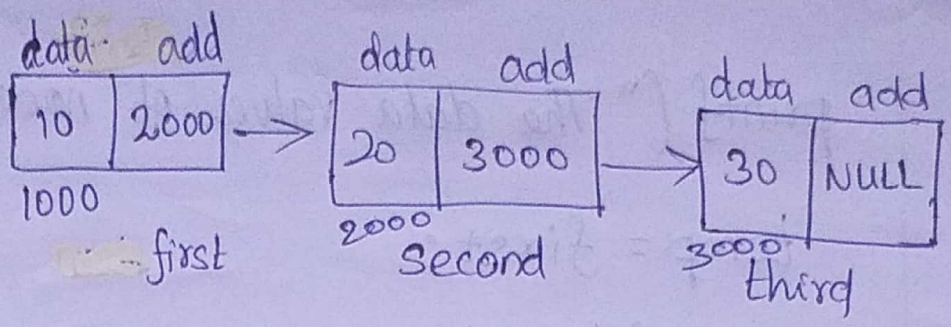
node.

data field:-

Data is a value that may be any data type such as int, char, float. etc.

address:-

This field contain the address of the next node. The last node contains "NULL".



8M (12) Example Program : Self Referential Str:-
(Linked list – Create New Nodes)

```
#include<stdio.h>
struct node
 {
 int data
 Struct node *add;
 } *first, *second, *third, *temp;
Void main()
 {
printf("Enter the data of node 1 \n");
 scanf("%d", &first->data);
 first->add = second;
printf(" Enter the data of node 2 \n");
 scanf("%d", &second->data);
 second->add = third;
printf("Enter the data of node 3 \n");
 Scanf("%d", &third->data);
 third->add = NULL;
```

```
printf("The data values of nodes are \n");
    temp = first;
    while(temp != NULL)
    {
    printf("Node value : %d ", temp->data);
    temp = temp->address;
    }
getch();
}
```

Output :-

```
Enter the data of node1
10
Enter the data of node2
20
Enter the data of node3
30
The data values of nodes are
Node value : 10
Node value : 20
Node value : 30
```

Structure:-

2M (IX)

✓ Structure in C language a user defined datatype that allows you to hold different type of elements.

✓ Each element of a structure is called member.

✓ It is widely used to store Student information, employee information, product information, book information etc.

Defining structure:-

2M (IX)

✓ The struct keyword is used to define structure.

✓ Let's see the syntax to define structure in C.

Syntax:-

```
struct structure_name
{
datatype member1;
datatype member2;
.
.
.
datatype member N;
};
```

Example:-

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

✓ Here struct the keyword, employee the tag name of structure. id, name, and salary are the members or fields of the structure.

Declaring structure variable:-

✓ we can declare variable for the structure, so that we can access the member

of Structure easily. There are two ways to declare structure variables.

1. By struct keyword within main() function.
2. By declaring variables at the time of declaring structure.

i) Example to declare structure variable by struct keyword. It should be declared within the main function.

```
struct employee
{
int id;
char name[50];
float salary;
};
struct employee e1,e2;
```

ii) To declare variable at the time of defining structure.

```
struct employee
{
int id;
char name[50];
float salary;
} e1,e2;
```

2M

(X) Accessing members of Structure:-

✓ There are two ways to access Structure members:

2M (X) 1. By .(member or dot operator)

2. By →(Structure pointer operator)

Example Program:-

```
#include<stdio.h>
 struct employee
  {
  char name[50];
  int wages;
  float days, bp, hra, np;
  };
 int main()
   {
    int n, i;
    printf("\n Enter the no. of employa");
    Scanf("%d", &n);
    Struct employee e[n];
  for(i=0; i<n; i++)
   {
  Scanf("
  printf("Enter the emp Name %d:", i+1);
```

```
scanf("%s", e[i].name);
printf("Enter the working days\n");
scanf("%f", &e[i].days);
printf("Enter the wages \n");
printf("%d", &e[i].wages);
e[i].bp = e[i].wages * e[i].days;
e[i].hra = e[i].bp * 0.12;
e[i].np = e[i].hra + e[i].bp;
}
for (i=0; i<n; i++)
{
printf("Employee name: %s\t", e[i].name);
printf("\n working days : %f \t", e[i].days);
printf("\n wages : %d \t", e[i].wages);
printf("\n Basic pay : %f \t", e[i].bp);
printf("\n Home Rent All : %f \t", e[i].hra);
printf("\n Net pay : %f \t", e[i].np);
}
return 0;
}
```

Output:

```
Enter the no. of employee
1
Enter the emp Name 1 : John
Enter the working days: 30
Enter the wages : 600
Employee Name : John
Working days : 30
wages : 600
Basic Pay : 18,000
Home Rent All : 2160
Net pay : 20160.
```

Nested structures:

✓ Nested structure in c language can have another structure as a member. There are two ways to define nested structure in c- language.

1. By separate structure
2. By Embedded structure

Separate structure:-

✓ We can Create 2 structures, but

dependent structure should be used inside the main structure as a member.

example:-

```
struct date
{
 int dd;
 int mm;
 int yyyy;
};
Struct employee
 {
 int id;
 char name[20];
 Struct date doj;
 } empl;
```

✓ As you can see, doj (date of joining is the variable of type date. Here doj is used as a member in employee structure.

✓ In this way, we can use data structure in many structures.

Embedded Structures:-

✓ we define structure within the structure also. It requires less code than previous way.

✓ But it cantbe used in many Strvctures.

example:-

```
struct employee
 {
 int id;
 char name[20];
 Struct date
   {
   int dd;
   int mm;
   int yyyy;
   } doj;
 }empl;
```

Accessing Nested Structure:-

✓ we can access the number of nested Structure by Outer Structure.

✓ Nested Structure member as given below

e1.doj.dd

e1.doj.mm

e1.doj.yyyy

8M (ii)

Example program for Nested Structure

to display the employee details?

program:-

```
#include<stdio.h>
 struct address
  {
   int pincode;
   char city[50];
   char area[50];
  };
 struct employee
  {
   int empid;
   float bs, ns, hra, da;
   char name[50];
   struct address add;
  }e;
 void main()
  {
  printf("Enter the employee details\n");
 printf("Enter the name\n");
 scanf("%s", &e.name);
printf("Enter the employee id\n");
 scanf("%d", &e.empid);
```

```
printf("Enter the basic Salary \n");
scanf("%f", &e.bs);
printf("Enter the pincode \n");
scanf("%d", &e.add.pincode);
printf("Enter the city \n");
scanf("%s", &e.add.city);
printf("Enter the area \n");
scanf("%s", &e.add.area);

printf("Enter the hra \n");
scanf("%f", &e.hra);
printf("Enter the da \n");
scanf("%f", &e.da);
e.ns = e.bs + e.hra + e.da;
printf("Employee details are \n");
printf("Name : %s \t", e.name);
printf("employee id : %d \t", e.empid);
printf("Net Salary : %f \t", e.ns);
printf("Basic Salary : %f \t", e.bs);
printf("Home Rent Allow : %f \t", e.hra);
printf("Daily Allowance : %f \t", e.da);
```

```
printf("City :%s", e.add.city);
printf("Area : %s", e.add.area);
getch();
}
```

Output:-

```
Enter the employee details
Enter the name
    karthick
Enter the employee id
    104019
Enter the basic salary
    24000
Enter the pincode
    600012
Enter the city
    xxxxxx
Enter the Area
    xxxxxxx
Enter the hra
    7000
Enter the da
    2000
Employee details are..
Name : karthick
employee id: 104019
```

Netsalary : 33000

Basic Salary : 24000

Home Rent Allow : 7000

Daily Allowance : 2000

Pincode : 600012

City : xxxxxx

Area : xxx xxx

JIT-2106