DATE : / 20
PAGE No

Draft

* Union :

- A union is a special data type that allows to store different data type in the same memory location
- a union can be defined with many members, but only one member can contain a value at any given time.
- Union provide an efficient way of using the same memory location for multiple purpose.
- Union can be created using union keyword.

Q. WAP to input following data of 30 employs of a Company.
(i) emp-code
(ii) emp_name
(iii) basic_salary.

Print the list of all the employs containing their name and and gross salary.

the gross salary can be calculated

Turbo C++ 4.0 Wind

DATE: / 20
PAGE No

by Gross_Salary = Basic_Salary + DA + HRA
DA = 110% of Basic Salary
HRA = 25 % of Basic salary

⇒
```
struct employe
  {
     int emp_code;
     char emp_name;
     float emp_salary;
  }e;
  Void main()
  {
     Printf(" Enter name");
```

14/19

Unit 6

Runtime or dynamic allocation

DATE : / 20
PAGE No

* Pointer :-

- A pointer is a variable that is used to store a memory address. The address is the location of another variable in memory.

- If one variable holds the address of another then it is said to point to the second variable

- Generally a pointer is declared to refer to a variable of its type

- Pointer Declaration :
  - data_type *ptr;
  - int *ptr;

* Pointer operators :

- & (Address operator) : reference operator
  - → Unary operator
  - → Return the address of its operand which must be a variable

Ex:
```
int a = 10;
int *iptr = &a;
```

iptr gets address of a & iptr point to a

DATE: / 20
PAGE No

* * Indirection (De-reference operator):

- Unary operator
- returns the value of the variable located at the address its operand stores.
- Complement or inverse of the address operator.

* Example:

```
int a = 10;
int *iptr = &a;
printf("%u", iptr); // Print add. of a
printf("%d", *iptr); // Print value of a.
```

Not Returning

DATE : / 20
PAGE No

15/4/19

* call by reference:

```
Void swap( int*, int* );
 void main
 {
     int a = 10, b = 20;
 Printf(" Before swapping a= %d and
     b= %d", a, b);
  swap (&a, &b);
 Printf(" After swapping a= %d and
          b= %d", a, b);
 }

 Void swap( int *Pa, int *Pb)
   {
       int temp;
       temp = *Pa;
       *Pa = *Pb;
       *Pb = temp;
   }
```

* Pointer Arithmetic:
  - following operators can be applied on pointers:

* Assignment (=)
* Arithmetic
  → Addition (+), increment (++)

*, /, % not applicable in pointer vars

DATE: / 20
PAGE No

- If a pointer is incremented by 1, the pointer will start pointing to the immediate next location.
- The value of the pointer will get increased by the size of the data type to which the pointer is pointing.

→ Subtraction (-), Decrement (--)

- If a pointer is decremented by 1, it will start pointing to the previous location.

* Relational :
==, !=, <, <=, >, >=

address is stored only Pointer

① Assignment

```
int a,b;
int *ptr1, *ptr2;
ptr1 = &a;
ptr2 = ptr1;
```

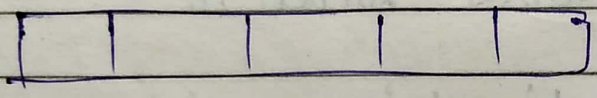
arr => address
*arr => value

DATE : / 20
PAGE No

* Pointers & array :

When an array is declared, Ban address i.e. address of the first element of the array is allocated.

- Example: ← Pointer variable
  int arr[5];
* Base address is stored in arr.
* arr is equal to &arr[0]
* * arr is equal to arr[0].



element	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
add	100	102	104	106	108
address	arr	arr+1	arr+2	arr+3	arr+4
values	*(arr)	*(arr+1)	*(arr+2)	*(arr+3)	*(arr+4)

Ex: Input

```
for (i=0 ; i<5 ; i++)
{
    Scanf ("%d", arr+i);
}
```

Print

```
for (i=0 ; i<5 ; i++)
{
    Printf("%d", *(arr+i));
}
```

Pointer to structure

DATE: / 20
PAGE No

26/4/19

* Self-Referential Structure:

- A self-Referential Structure is one which refers to or point to another structure of the same type.
- Used for implementing Linked List.
- Syntax:

```
struct struct_Name
{
Data-type variable_name;
struct struct_name *pointer_name;
};
```

Example:

```
struct linked_list
{
    int data;
    struct linked_list *next;
};
```

* Linked List:

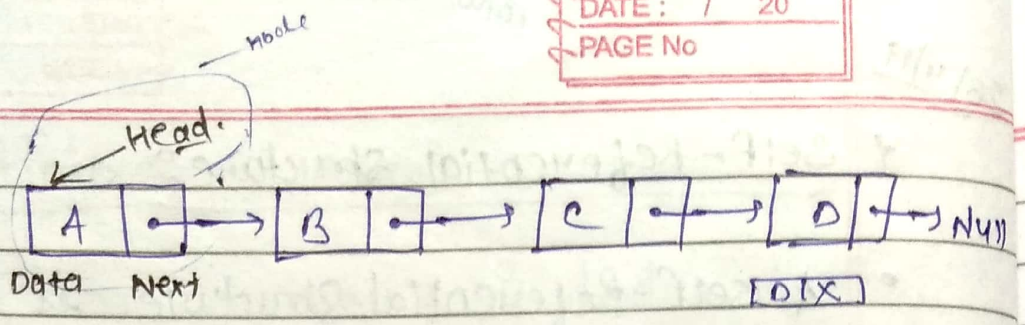
1→2→3

- Linked list is a linear data structure
- Unlike arrays, linked list elements are not stored at contiguous location.
- The elements are linked using pointers.

free => node delete

malloc → Run time memory allocate

DATE : / 20
PAGE No

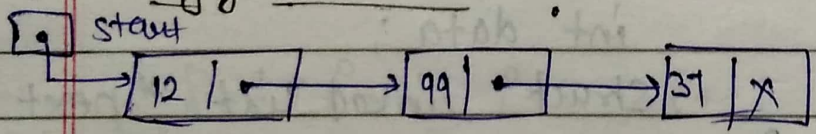


* Advantages:
  - Dynamic size
  - ease of insertion / deletion

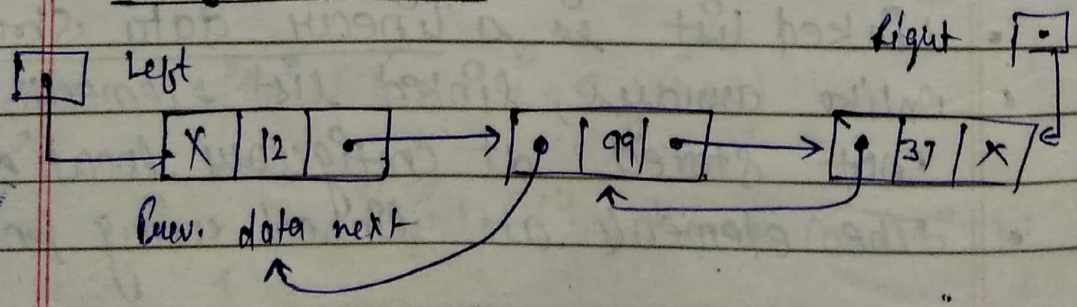
* Disadvantages:
  - Random access is not allowed
  - use more memory than array
  - Difficult to traverse in reverse order.
  - More time required to access individual elements.

* Linked List type:

1) Singly Linked list:



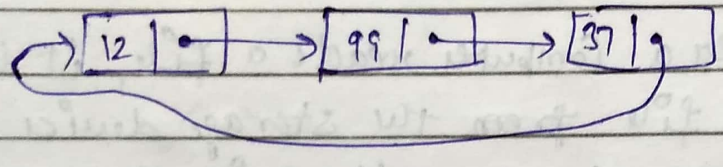
(2) doubly Linked List:



DATE: / 20
PAGE No

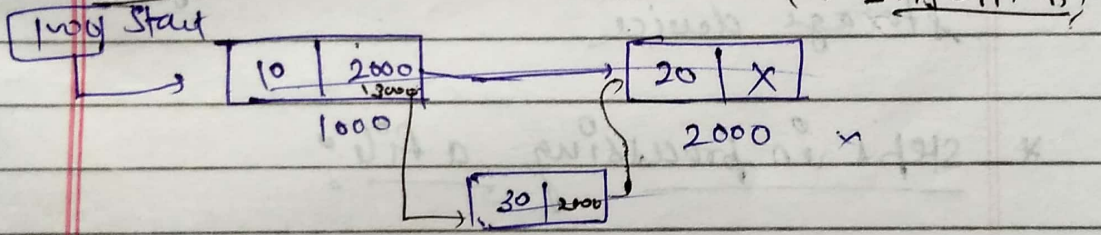
```
structure dblist
{
    int data;
    struct dblist *prev, *next;
};
```

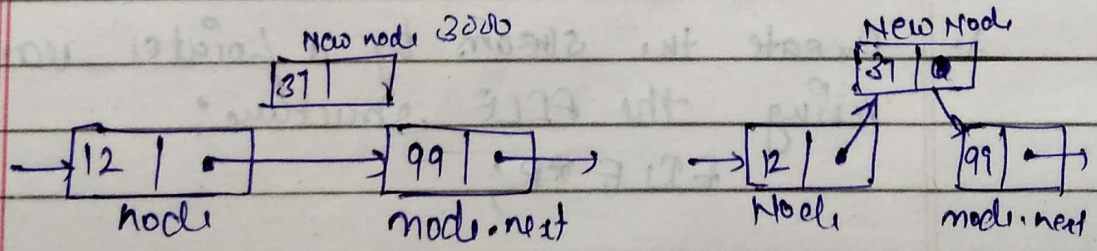
(3) Circular Linked List :-



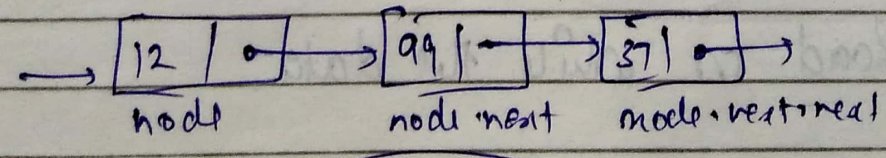
* Linked list operation :-

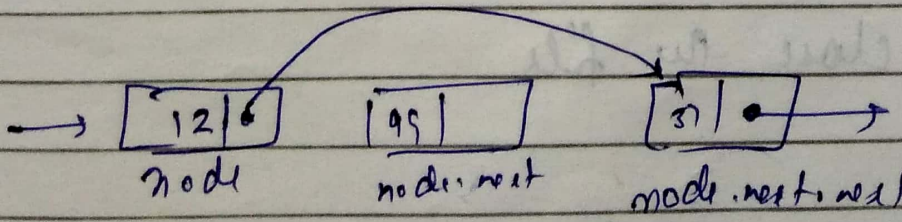
(1) Insertion :





(2) Deletion :





FILE

DATE : / 20
PAGE No

* File handling:

- A file is a collection of related data that a computer treats as a single unit.

- The contents of files remain intact when a computer shuts down.

- When a computer reads a file, it copies the file from the storage device to memory; when it writes to a file, it transfers data from memory to the storage device.

* Steps in processing a file:

1. Create the stream by a pointer variable using the FILE structure:
   FILE *P;

2) Open the file, associating the stream name with the file name.

3) Read or write the data.

4. Close the file.