Linked List

❖ 4.1 Pointers Revision

- ✓ Pointer: A pointer is a variable which contains an address of another variable in memory.
- ✓ Declared by * indicator..
- ✓ We can create a pointer variable in C using
- ✓ following syntax:
  Declaration:int*ptr

❖ 4.2 Revision of Structure

- ✓ Structures hold data that belong together.
  Examples:
- ✓ Student record: student id, name, major,gender, start year, …
- ✓ Bank account: account number, name,currency, balance, …
- ✓ Address book: name, address, telephone number, …
- ✓ In database applications, structures are called records.

Definition of a structure:

```
struct <struct-type>{
<type> <identifier_list>;
<type> <identifier_list>;
...
} ;
```

Example:

```
struct Date {
int day;
int month;
int year;
} ;
```

❖ 4.3 Revision of structure using pointers

- ✓ Pointers are symbolic representation of addresses.
- ✓ Pointers enable programs to simulate call-by reference and to create and manipulate dynamic data structures.
- ✓ Referencing a value through a pointer is called indirection.
- ✓ The * is called indirection operator.

❖ 4.4 Dynamic Memory Allocation

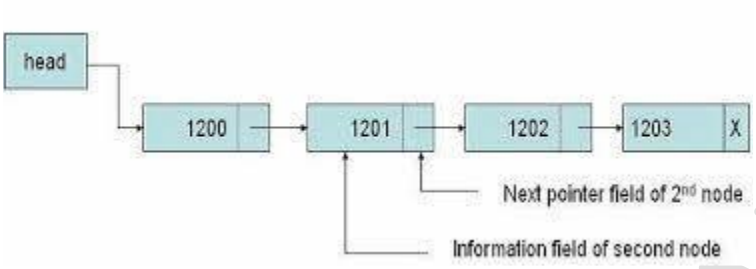
- ✓ In the Dynamic memory allocation , the memory is allocated to a variable or program at the run Time.
- ✓ The only way to access this dynamically allocated memory is through pointer.

Types of dynamic memory allocation:

1. Malloc ( )
2. Calloc( )

3. Realloc( )
4. Free ( )

❖ 4.5 Linked list Presentation



- ✓ A singly linked list is to expand each node to contain a link or pointer to the next node. This is also called as a one way chain.
- ✓ First contain the address of the first node of the lists.
- ✓ Each node in the list consist of two parts::
- ✓ 1.. Information(INFO)
- ✓ 2... Address or printer to next node (LINK).

❖ 4.6 Types of Linked List

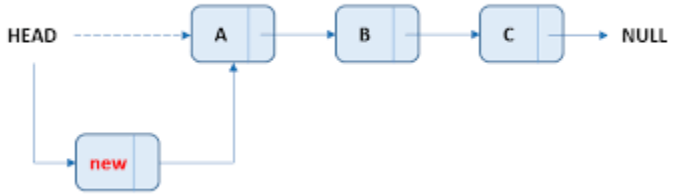
- ✓ Singly Linked List
- ✓ Singly Circular Linked List
- ✓ Doubly Linked List
- ✓ Doubly Circular Linked List
- ✓ Ordered Linked List

❖ 4.7 Basic operations on singly linked list

The basic operation for linked list is:

- ✓ To create a linked list.
- ✓ Traversing a linked list.
- ✓ Insert New node at beginning (at first)
- ✓ Insert new node at end
- ✓ Insert new node at any location or in between the list
- ✓ Inserting a node in to an ordered linear list.
- ✓ Delete a first node (at beginning)
- ✓ Delete a last node (at end)
- ✓ Delete a node on basis of node number
- ✓ Searching element in linked list Count the number of nodes in linked list
- ✓ Count the number of nodes in linked list.

- **ALGORITHMS FOR SINGLY LINKED LIST BEGINNING:**



Step 1: If AVAIL=NULL then

Write "Availability Stack is Empty"

Else

NEW_NODE=AVAIL

AVAIL = AVAIL->LINK

Step 2: If FIRST = NULL then

NEW_NODE -> INFO = X

NEW_NODE -> LINK = NULL

FIRST = NEW_NODE

Else

NEW_NODE -> INFO = X

NEW_NODE -> LINK = FIRST

FIRST = NEW_NODE

Step 3: Exit

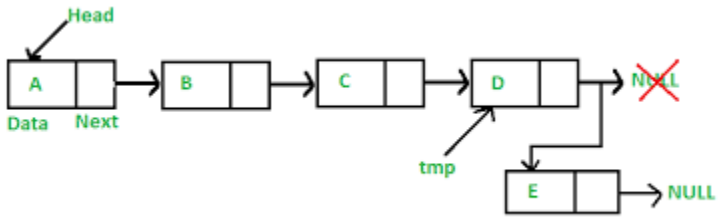
NEW_NODE -> INFO = X

NEW_NODE -> LINK = FIRST

FIRST = NEW_NODE

Step 4: Exit

- **ALGORITHM TO INSERT NEW NODE AT END OF LINKED LIST**



Step 1: If AVAIL=NULL then

Write "Availability Stack is Empty"

Else

NEW_NODE=AVAIL

AVAIL = AVAIL->LINK

Step 2: If FIRST = NULL then

NEW_NODE -> INFO = X

NEW_NODE -> LINK = NULL

FIRST = NEW_NODE

Else

NEW_NODE -> INFO = X

NEW_NODE -> LINK = NULL

SAVE = FIRST

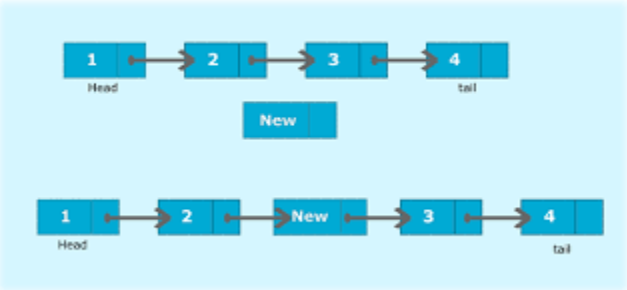
Repeat while SAVE->LINK ≠ NULL

SAVE = SAVE->LINK

SAVE->LINK = NEW_NODE

Step 3: Exit

- **ALGORITHM TO INSERT NEW NODE AT SPECIFIC LOCATION SINGLE LINK LIST**



Step 1: If AVAIL=NULL then

Write “Availability Stack is Empty”

Else

NEW_NODE=AVAIL

AVAIL = AVAIL->LINK

Step 2: If FIRST = NULL then

Write “Specified Node Not Found”

Else

NEW_NODE -> INFO = VALUE

SAVE = FIRST

Repeat while X ≠ SAVE->INFO and SAVE->LINK ≠ NULL

PRED = SAVE

SAVE = SAVE->LINK

If X = SAVE->INFO then

NEW_NODE->LINK= SAVE->LINK

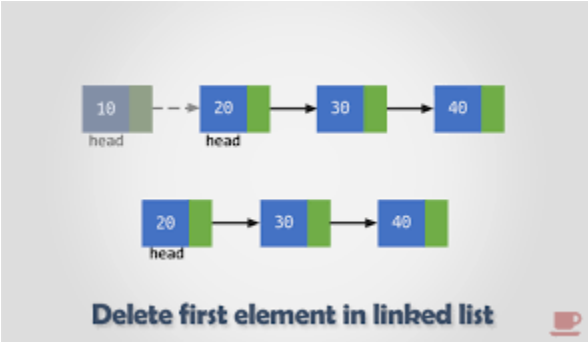
SAVE->LINK=NEW_NODE

Else

Write “Specified Node Not Found”

Step 3: Exit

- **DELETE FIRST NODE FROM SINGLE LINKED LIST**



Step 1: If FIRST = NULL then

Write “Linked List is Empty”

Step 2: If FIRST->LINK = NULL then

Return FIRST->INFO

FIRST=NULL

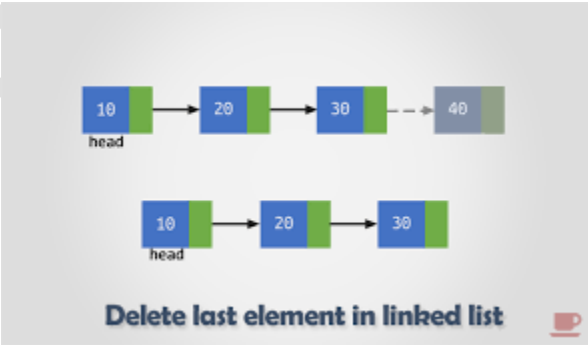
Else

Return FIRST->INFO

FIRST=FIRST->LINK

Step 3: Exit

- **ALGORITHM TO DELETE LAST NODE FROM SINGLE LINKED LIST**



Step 1: If FIRST = NULL then

Write “Linked List is Empty”

Step 2: If FIRST->LINK = NULL then

Return FIRST->INFO

FIRST=NULL

Else

SAVE=FIRST

Repeat while SAVE->LINK ≠ NULL

PRED=SAVE

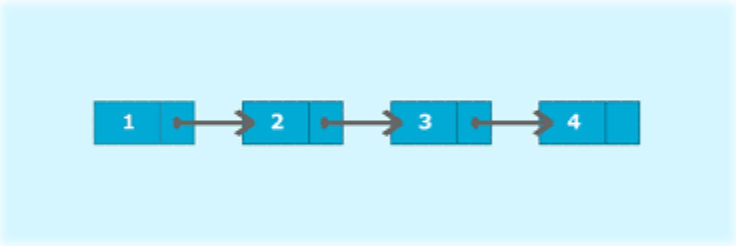
SAVE=SAVE->LINK

Return SAVE->INFO

PRED->LINK=NULL

Step 3: Exit

- **ALGORITHM TO SEARCH NODE IN SINGLE LINKED LIST**



Step 1: FLAG = 0

SAVE=FIRST

Step 2: Repeat step 3 while SAVE ≠ NULL

Step 3: If SAVE->INFO = X then

FLAG = 1

SAVE=SAVE->LINK

Else

SAVE=SAVE->LINK

Step 4: If FLAG = 1 then

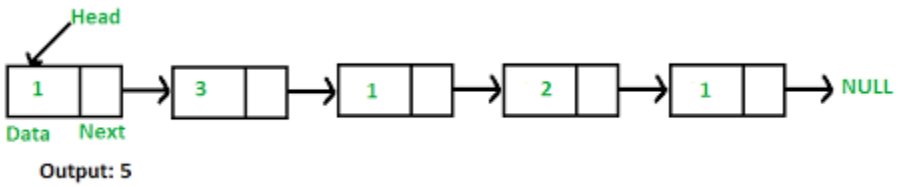
Write “Search is Successful”

Else

Write “Search is not successful”

Step 5: Exit

- **ALGORITHM COUNT NUMBER OF NODES IN SINGLE LINKED LIST**



Step 1: Count = 0

SAVE = FIRST

Step 2: Repeat step 3 while SAVE ≠ NULL

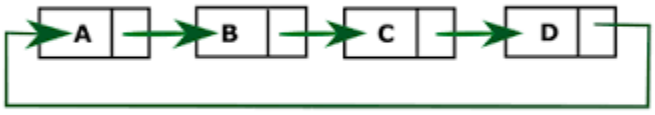
Step 3: Count= Count + 1

SAVE=SAVE->LINK

Step 4: Return Count

❖ 4.8 Concepts of circular linked list

- ✓ A list in which last node contains a link or pointer to the first node in the list is known as Circular linked list.
- ✓ Representation of circular linked list is shown below:



- ✓ In a circular linked list there are two methods to know if a node is the first node or not.

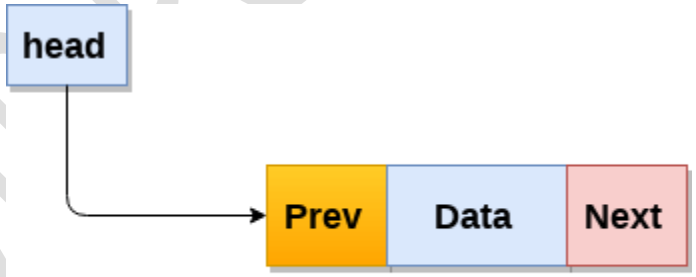
- ✓ Either a external pointer, list, points the first node or A header node is placed as the first node of the circular list.
- ✓ The header node can be separated from the others by either heaving a sentinel value as the Info part or having a dedicated flag variable to specify if the node is a header node or not.

❖ 4.9 Difference between circular linked list and singly linked list

- ✓ Every node is accessible from a given node, that is from this given node be reached by chaining through the list.
- ✓ To delete a node from a singly linked list, it is required to have the first node address. Such a requirement does not exist for a circular linked list.
- ✓ Certain operations such as splitting, concatenation, becomes more efficient in circular link list.
- ✓ The disadvantage of a circular list is, without some care in processing, it is possible to get into an infinite loop !.....
- ✓ In circular linked list, the detection of the end by placing a special node easily identified in the circular list is called a list head.

❖ 4.10 Doubly linked list: Representation

- ✓ Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.
- ✓ Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer), and pointer to the previous node (previous pointer).
- ✓ A sample node in a doubly linked list is shown in the figure.



Node

Doubly Linked List

❖ 4.11 Difference between Doubly linked list and singly linked list

❖ 4.12 Applications of the linked list

- ✓ Polynomial representation, automatic polynomial manipulations are performed by link list.
- ✓ Addition and subtractions operations of polynomial are easily implemented using link list.
- ✓ Symbol table creation.
- ✓ Multiple precision arithmetic and representation of sparse matrices.
- ✓ The polynomial equation is algebraic expression which is used in scientific and business applications.