|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| <hitle> | LinkedList | <chare> | 1 | <pext> | Linked List can be defined as collection of objects called nodes  that are randomly stored in the memory.  A node contains two fields like data - stored at that particular address  and the pointer - contains the address of the next node in the memory.  The last node of the list contains pointer to the null.  -----------types of linked lists-------------  1. Singly Linked List − The nodes only point to the address of the next node in the list.  2. Doubly Linked List − The nodes point to the addresses of both previous and next nodes.  3. Circular Linked List − The last node in the list will point to the first node in the list. It can either be singly linked or doubly linked.  ----------linked list pros over array-------------  1. Array contains some of limitations:  1-1. The size of array must be known in advance  before using it in the program.  1-2. Increasing size of the array is a time taking process.  It is almost impossible to expand the size of the array at run time.  1-3. All the elements in the array need to be  contiguously stored in the memory.  Inserting any element in the array needs shifting of all its predecessors.  2. Linked list is the data structure which can overcome  all the limitations of an array.  2-1. It allocates the memory dynamically.  All the nodes of linked list are non-contiguously stored in the memory  and linked together with the help of pointers.  2-2. Sizing is no longer a problem  since we do not need to define its size at the time of declaration.  List grows as per the program's demand  and limited to the available memory space. | </end> |
| <hitle> | UpdatingRecords | <chare> | 2 | <pext> | UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords | </end> |
| <hitle> | UpdatingRecords | <chare> | 2 | <pext> | UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords  UpdatingRecords | </end> |