Data are a set of values. A data item refers to a single unit of values. Data items that are divided into sub items are called group items, those that are not divided are called elementary items. Information means processed data.

Data structure: A logical or mathematical model of organization of data. It includes the organization of data , operations performed on data and the memory requirements.

Broadly classified into :

Primitive DS – That is already defined in the system . int,char , real numbers

Non-primitive- Which is not defined. Further classified into linear and non-linear.

Linear- Relationship between the elements is in a linear pattern. Arrays,Linked list,stack and queues

Non-linear- Has a hierarchical relationship between the elements. Trees and graphs.

Linear Data Structure operations:

1. Traversal- Accessing each element exactly once.
2. Search- Finding the location of an element.
3. Insertion- Adding a new element
4. Deletion- Removing an element.
5. Sorting- Arranging the elements in ascending or descending order
6. Merging- Combining two lists into a single list.

Linear Arrays

A linear array is a list of finite number of homogeneous elements stored in continuous memory location under a common name. The elements of the array are referenced respectively by an index set consisting of n consecutive numbers.

The number of elements n is called the length or size of the array. The number k in A[K] is called the subscript and A[k] is called the subscripted variable.

Length= Upperbound-Lowebound + 1

Let LA be a linear array.

LOC(LA[k])= Address of the element LA[k] of the array LA, Address of the first element of LA-BASE(LA).

**LOC(LA[k])= BASE(LA)+w(k-lowerbound), where w is the number of words per memory cell for the array LA.**

Insertion and Deletion in arrays

Let A be a collection of data elements. Inserting refers to the operation of adding another element to the collection A and deleting refers to the operation of removing one of the elements from A.

Inserting an element at the end of a linear array can be easily done provided the memory space allocated for the array is large enough to accommodate the additional element. On the other hand suppose we need to insert an element in the middle of the array then on the average half of the elements must be moved downwards to new location to accommodate the new element and keep the order of other elements.

Similarly deleting an element at the end of an array is not difficult but deleting an element somewhere in the middle of the array would require that each subsequent element be move one location upward in order to fill up the array.

Dynamic memory allocation

The process of allocating memory at runtime is known as dynamic memory allocation.

Dynamic memory management techniques permit us to allocate additional memory space or release unwanted space at runtime thus optimizing the use of storage space. The free storage area is known as heap.

Allocating a block of memory

A block of memory may be allocated using the function malloc. The malloc function reserves a block of memory of specified size and returns a pointer of type void. This means that we can assign it to any type of pointer.

Ptr=(cast\_type\*) malloc(byte\_size);

Ptr is pointer of the type cast type. The malloc returns a pointer to an area of memory with size byte\_size.

The storage space allocated dynamically has no name and therefore its contents can be accessed only through a pointer. Malloc allocates a block of contiguous bytes. If it fails to find the block then it returns a NULL.

Allocating multiple blocks of memory

Calloc is another memory allocation function that is normally used for requesting memory space at runt ime for storing derived data types such as arrays and structures. While malloc allocates a single block of storage space calloc allocates multiple blocks of storage each of the same size and then sets all bytes to zero.

Ptr=(cast\_type\*)calloc(n,elem\_size);

The above statement allocates a contiguous space for n blocks each of size elem\_size bytes. All bytes are initialized to zero and a pointer to the first byte of allocated region is returned. If there is not enough space a null pointer is returned.

Releasing the Used space

When we no longer need the data we stored in a block of memory and we do not intend to use that block for storing any other information we may release that block of memory for future use using the free function.

free(ptr);

ptr is a pointer to a memory block which has already been created by malloc or calloc.

Altering the size of a block

It is likely that we discover later the previously allocated memory is not sufficient and we need additional space for more elements. It is also possible that the memory allocated is larger than necessary and we want to reduce it. In both the cases we can change the memory size already allocated with the help of the function realloc. This process is called the reallocation of memory.

ptr=(cast\_type\*)malloc(size);

ptr=realloc(ptr,newsize);

This function allocates a new memory space of size new size to the pointer variable ptr and returns a pointer to the first byte of the new memory block. The new size may be larger or smaller than the size. The new block may or may not begin at the same place as the old one. In case it is not able to find additional space in the same region it will create the same in an entirely new region and move the contents of the old block with the new block.

Linked List

A linked list or a one-way list is a linear collection of data elements called nodes where the linear order is given by means of pointers. That is each node is divided into two parts the first part contains the information of the element and the second part called the link field contains the address of the next node in the list.

HEAD/START

Info link

Info link

Info link

info

The pointer of the last node contains a special value called the null pointer. The linked list also contains a list pointer variable called HEAD or STARTS which contains the address of the first node in the list.

Advantages of linked list

A linked list is a dynamic data structure. Therefore, the primary advantage of linked lists over arrays is that linked list can grow or shrink in size during the execution of a program.

Another advantage is that a linked list does not waste memory space. It uses the memory that is just needed for the list at any point of time. This is because it is not necessary to specify the number of nodes to be used in the list.

The third and the most important advantage is that the linked list provides flexibility in allowing the items to be rearranged efficiently. It is easier to insert or delete items by rearranging the links.

The major limitation of the linked list is that the access to any arbitrary item is little cumbersome and time consuming. Whenever we deal with a fixed length list it would be better to use an array rather than a linked list. We must also note that a linked list will use more storage than an array with the same number of items. This is because each item has an additional link field.

Doubly Linked list/Two-Way list

Two way lists can be traversed in two directions-in the usual forward direction from the beginning of the list to the end or in the backward direction from the end of list to the beginning.

A two way list is a linear collection of data element called nodes where each node N is divided into three parts:

1. An information field INFO which contains the data of N
2. A pointer field FORW which contains the location of the next node in the list
3. A pointer field BACK which contains the location of the preceding node in the list

The list also requires two list pointer variables FIRST which points to the first node in the list and LAST which points to the last node in the list.

The forw pointer of the first node and the back pointer of the last node is NULL.

Stack

A stack is a list of elements in which an element may be inserted or deleted only at one end called the top of the stack.

Two basic operations associated with stack:

1. “PUSH” is the term used to insert an element into a stack
2. “POP” is the term used to delete an element from a stack.

The last item added to a stack is the first item to be removed. Stack is also known as a LIFO data structure.

Memory overflow

Sometimes new data are to be inserted into a data structure but there is no available space, that is free storage list is empty. This situation is usually called memory overflow.

Memory Underflow

The term underflow refers to the situation where one wants to delete data from a data structure that is empty.

Queues

A queue is a linear list of elements in which deletions can take place only at one end called the front and insertions can take place only at the other end called the rear. The terms front and rear are used in describing a linear list only when it is implemented as a queue.

Queues are also called first-in-first-out (FIFO) lists, since the first element in a queue will be the first element out of the queue. Queues abound in everyday life. The automobiles waiting to pass through an intersection form a queue, in which the first car in line is the first car through; the people waiting in line at a bank form a queue, where the first person in line is the first person to be waited on and so on. An important example of a queue in computer science occurs in a timesharing system, in which programs with the same priority form a queue while waiting to be executed.

Algorithms

An algorithm is a finite set of instructions that if followed accomplishes a particular task. All algorithms must satisfy the following criteria:

1. Input- There are zero or more quantities that are externally supplied
2. Output- At least one quantity is produced
3. Definiteness- Each instruction is clear and unambiguous
4. Finiteness- If we trace out the instructions of an algorithm then for all cases the algorithm terminates after a finite number of steps
5. Effectiveness- Every instruction must be basic enough to be carried out, in principle, by a person using only pencil and paper; it also must be feasible