Shishir Shetty Div 3

Roll no:-41

Experiment no 6:

Aim: Implementation of Singly Linked List

Objective : It is used to implement stacks and queue which are linked needs throughout computer science .To prevent the Collision between the data in the Hash map.we use a singly Linked list

Theory ;

A singly linked list is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence. A singly linked list allows traversal of data only in one way.

Traversal in a Singly Linked List

Traversing a linked list means accessing the nodes of the list in order to perform some processing on them. In the code below, just the data of the node is printed.

Insertion of a node

In order to insert a node at the beginning, a new node is created which is assigned data and a pointer which references the first node. The start is then changed to the new node.

Deletion of a node

The start pointer is changed to the next pointer of the first node. The first node is then deleted using the free function.

Algorithm

Insertion of a node

Step 1: IF AVAIL = NULL Write OVERFLOW Go to Step 7 [END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL NEXT

Step 4: SET DATA = VAL Step 5: SET NEW\_NODE NEXT = START Step 6: SET START = NEW\_NODE

Step 7: EXIT

Deletion of a node

Step 1: IF START = NULL Write UNDERFLOW Go to Step 5 [END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START NEXT

Step 4: FREE PTR

Step 5: EXIT

Traversal in a Singly Linked List

Step 1: [INITIALIZE] SET PTR = START

Step 2: Repeat Steps 3 and 4 while PTR != NULL

Step 3: Apply Process to PTR DATA

Step 4: SET PTR = PTR NEXT [END OF LOOP]

Step 5: EXIT

Code

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

};

struct node \*start = NULL;

struct node \*display(struct node \*);

struct node \*insert\_beg(struct node \*);

struct node \*delete\_beg(struct node \*);

int main() {

int option;

do {

printf("\n\n \*\*\*\*\*MAIN MENU \*\*\*\*\*");

printf("\n 1: Add a node at the beginning");

printf("\n 2: Display the list");

printf("\n 3: Delete a node from the beginning");

printf("\n\n Enter your option : ");

scanf("%d", &option);

switch(option) {

case 1: start = insert\_beg(start);

break;

case 2: start = display(start);

break;

case 3: start = delete\_beg(start);

break;

default: printf("Enter valid choice");

break;

}

} while(option != 13);

getch();

return 0;

}

struct node \*insert\_beg(struct node \*start) {

struct node \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node->data = num;

new\_node->next = start;

start = new\_node;

return start;

}

struct node \*display(struct node \*start) {

struct node \*ptr;

ptr = start;

while(ptr != NULL) {

printf("\t %d", ptr->data);

ptr = ptr->next;

}

return start;

}

struct node \*delete\_beg(struct node \*start) {

struct node \*ptr;

ptr = start;

start = start->next;

free(ptr);

return start;

}

The syntax for creating a node

struct Node

{

int Data;

Struct Node \*next;

};

Insertion of a node

void insertStart (struct Node \*\*head, int data)

{

struct Node \*newNode = (struct Node \*) malloc (sizeof (struct Node));

newNode - >

data = data;

newNode - >

next = \*head;

//changing the new head to this freshly entered node

\*head = newNode;

}

Deletion of a node

void deleteStart(struct Node \*\*head)

{

struct Node \*temp = \*head;

// if there are no nodes in Linked List can't delete

if (\*head == NULL)

{

printf ("Linked List Empty, nothing to delete");

return;

}

// move head to next node

\*head = (\*head)->next;

free (temp);

}

Traversal in a Singly Linked List

void display(struct Node\* node)

{

printf("Linked List: ");

// as linked list will end when Node is Null

while(node!=NULL){

printf("%d ",node->data);

node = node->next;

}

printf("\n");

}

Output



Conclusion :

* The linked list data structure has been successfully implemented.
* Linked list allows dynamic memory allocation and thus works well for saving data whose size is not known.
* Linked list requires more memory than a simple array.