**Data Structure lab**

**Operations on Singly Link list**

**Megha Patel**

**20124107**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node {

int data; // Data

struct node \*next;

int info;

struct node\* link;

}\*start;

void createList()

{

int n;

printf("how many nodes you want to enter \n");

scanf("%d", &n);

struct node \*newNode, \*temp;

int data, i;

start = (struct node \*)malloc(sizeof(struct node));

// Terminate if memory not allocated

if(start == NULL)

{

printf("Unable to allocate memory.");

exit(0);

}

printf("now enter them \n");

scanf("%d", &data);

start->info = data; // Link data field with data

start->link = NULL; // Link address field to NULL

// Create n - 1 nodes and add to list

temp = start;

for(i=2; i<=n; i++)

{

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

/\* If memory is not allocated for newNode \*/

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

scanf("%d", &data);

newNode->info = data; // Link data field of newNode

newNode->link = NULL; // Make sure new node points to NULL

temp->link = newNode; // Link previous node with newNode

temp = temp->link; // Make current node as previous node

}

}

void traverse()

{

struct node\* temp;

// List is empty

if (start == NULL)

printf("\nList is empty\n");

// Else print the LL

else {

temp = start;

while (temp != NULL) {

printf(" %d\n",temp->info);

temp = temp->link;

}

}

}

void insertAtFront()

{

int data;

struct node\* temp;

temp = malloc(sizeof(struct node));

printf("\nEnter number to be inserted at starting : ");

scanf("%d", &data);

temp->info = data;

// Pointer of temp will be

// assigned to start

temp->link = start;

start = temp;

}

void insertAtEnd()

{

int data;

struct node \*temp, \*head;

temp = malloc(sizeof(struct node));

// Enter the number

printf("\nEnter number to"

" be inserted at end: ");

scanf("%d", &data);

// Changes links

temp->link = 0;

temp->info = data;

head = start;

while (head->link != NULL)

{

head = head->link;

}

head->link = temp;

}

void insertAtPosition()

{

struct node \*temp, \*newnode;

int pos, data, i = 1;

newnode = malloc(sizeof(struct node));

// Enter the position and data

printf("\nEnter position and data to be inserted:");

scanf("%d %d", &pos, &data);

// Change Links

temp = start;

newnode->info = data;

newnode->link = 0;

while (i < pos - 1) {

temp = temp->link;

i++;

}

newnode->link = temp->link;

temp->link = newnode;

}

void deleteFirst()

{

struct node\* temp;

if (start == NULL)

printf("\nList is empty\n");

else {

temp = start;

start = start->link;

free(temp);

}

}

void deleteEnd()

{

struct node \*temp, \*prevnode;

if (start == NULL)

printf("\nList is Empty\n");

else {

temp = start;

while (temp->link != 0) {

prevnode = temp;

temp = temp->link;

}

free(temp);

prevnode->link = 0;

}

}

void deletePosition()

{

struct node \*temp, \*position;

int i = 1, pos;

// If LL is empty

if (start == NULL)

printf("\nList is empty\n");

// Otherwise

else {

printf("\nEnter index to be deleted: ");

// Position to be deleted

scanf("%d", &pos);

position = malloc(sizeof(struct node));

temp = start;

// Traverse till position

while (i < pos - 1) {

temp = temp->link;

i++;

}

// Change Links

position = temp->link;

temp->link = position->link;

// Free memory

free(position);

}

}

void reverseList()

{

struct node \*t1, \*t2, \*temp;

t1 = t2 = NULL;

// If LL is empty

if (start == NULL)

printf("List is empty\n");

// Else

else {

// Traverse the LL

while (start != NULL) {

// reversing of points

t2 = start->link;

start->link = t1;

t1 = start;

start = t2;

}

start = t1;

// New head Node

temp = start;

printf("Reversed linked "

"list is : ");

// Print the LL

while (temp != NULL) {

printf("%d ", temp->info);

temp = temp->link;

}

}

}

int main()

{

int choice;

while (1)

{

printf("\n\t1 To see list\n");

printf("\t2 insertion at starting\n");

printf("\t3 insertion at end\n");

printf("\t4 For insertion at any position\n");

printf("\t5 For deletion first element\n");

printf("\t6 For deletion last element\n");

printf("\t7 For deletion of element at any position\n");

printf("\t8 To reverse the linked list\n");

printf("\t9 To exit\n");

printf("\nEnter Choice :\n");

scanf("%d", &choice);

switch (choice) {

case 1:

createList();

printf("traverse of list is\n");

traverse();

break;

case 2:

createList();

insertAtFront();

printf("after inserting at front\n");

traverse();

break;

case 3:

createList();

insertAtEnd();

printf("after inserting at end\n");

traverse();

break;

case 4:

createList();

insertAtPosition();

printf("after inserting at given position\n");

traverse();

break;

case 5:

createList();

deleteFirst();

printf("after deleting at front\n");

traverse();

break;

case 6:

createList();

deleteEnd();

printf("after deletion at end\n");

traverse();

break;

case 7:

createList();

deletePosition();

printf("after eletion at given postion\n");

traverse();

break;

case 8:

createList();

reverseList();

break;

case 9:

exit(1);

break;

default:

printf("Incorrect Choice\n");

}

}

return 0;

}

