**Program to implement Singly Linked List.**

#include <stdio.h>

#include <stdlib.h>

void create();

void display();

void insertBegin();

void insertEnd();

void insertPosi();

void deleteBegin();

void deleteEnd();

void deletePosi();

struct node {

int data;

struct node \*next;

};

struct node \*head = NULL;

int main() {

int choice = 0;

printf("====== MENU ======\n");

printf("1. CREATE\n");

printf("2. DISPLAY\n");

printf("3. Insert in the beginning.\n");

printf("4. Insert at the end.\n");

printf("5. Insert at a Position\n");

printf("6. Delete at the beginning.\n");

printf("7. Delete at the End.\n");

printf("8. Delete a specific.\n");

printf("9. EXIT.\n");

while (choice != 9) {

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: create();

break;

case 2: display();

break;

case 3: insertBegin();

break;

case 4:

insertEnd();

break;

case 5: insertPosi();

break;

case 6: deleteBegin();

break;

case 7: deleteEnd();

break;

case 8: deletePosi();

break;

case 9: printf("Exiting...\n");

break;

default: printf("Wrong Choice!!!");

break;

}

}

return 0;

}

void create(){

struct node \*newnode, \*temp;

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

printf("enter the data value of the new node: ");

scanf("%d", &newnode -> data);

newnode -> next = NULL;

if(head == NULL){

head = newnode;

}else{

temp = head;

while(temp-> next != NULL){

temp = temp -> next;

}

temp -> next = newnode;

}}

void display(){

struct node \*ptr;

if(head == NULL){

printf("The link list is empty");

}else{

ptr = head;

printf("Nodes in the list are: ");

while(ptr != NULL){

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

ptr = ptr -> next;

}

printf("\n");

}}

void insertBegin(){

struct node \*newnode, \*ptr;

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

printf("enter the data value of the newnode: ");

scanf("%d", &newnode -> data);

newnode -> next = NULL;

if(head == NULL){

head = newnode;

}else{

newnode -> next = head;

head = newnode;

}

}

void insertEnd(){

struct node \*newnode, \*ptr;

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

printf("enter the data value of the newnode: ");

scanf("%d", &newnode -> data);

newnode -> next = NULL;

if(head == NULL){

head = newnode;

}else{

ptr = head;

while(ptr -> next != NULL){

ptr = ptr -> next;

}

ptr -> next = newnode;

}

}

void insertPosi(){

struct node \*newnode, \*temp, \*ptr;

int count = 0, posi = 0;

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

printf("enter the position of the new node(index starts from 0): ");

scanf("%d", &posi);

printf("enter the data value of the newnode: ");

scanf("%d", &newnode -> data);

newnode -> next = NULL;

if(posi == 0){

newnode -> next = head;

head = newnode;

return;

}else{

temp = head;

for(count = 1; count < posi; count++){

temp = temp -> next;

if(temp == NULL){

printf("INVALID position");

return ;

}

}

newnode -> next = temp -> next;

temp -> next = newnode;

}

}

void deleteBegin(){

struct node \*ptr;

if(head == NULL){

printf("The link list is Empty\n");

return;

}else{

ptr = head;

head = head -> next;

printf("The deleted node value is: %d\n", ptr -> data);

free(ptr);

}

}

void deleteEnd(){

struct node \*ptr, \*temp;

if(head == NULL){

printf("The link list is empty.\n");

return;

}else if(head -> next == NULL){

ptr = head;

head = NULL;

printf("The deleted element node is %d\n", ptr -> data);

free(ptr);

}else{

ptr = head;

while(ptr -> next != NULL){

temp = ptr;

ptr = ptr -> next;

}

temp -> next = NULL;

printf("The deleted element is %d\n", ptr -> data);

free(ptr);

}}

void deletePosi(){

int count = 0, posi = 0;

struct node \*ptr, \*temp;

if(head == NULL){

printf("The link list is Empty.\n");

return;

}else{

printf("Enter the position of the node to be deleted(index starts at 0): ");

scanf("%d", &posi);

ptr=head;

if(posi == 0){

head = head -> next;

printf("The deleted node value is %d\n", ptr -> data);

free(ptr);

}else{

for(count = 0; count < posi; count++){

temp = ptr;

ptr = ptr -> next;

if(ptr == NULL){

printf("INVALID position.\n");

return;

}}

temp -> next = ptr-> next;

printf("The deleted node value is %d\n", ptr -> data);

free(ptr);

}}

}

**Output:**



