Linked List code:Delete &Display

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

void create\_ll(struct node \*\*start);

void display(struct node \*start);

void pop(struct node \*\*start);

void end\_delete(struct node \*\*start);

void delete\_at\_pos(struct node \*\*start);

void free\_list(struct node \*start);

int main(void)

{

struct node \*start = NULL;

int option;

do

{

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

printf("\n 1: Create a list");

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

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

printf("\n 4: Delete a node from the end");

printf("\n 5: Delete a from a specific position");

printf("\n 6: EXIT");

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

scanf("%d", &option);

switch (option)

{

case 1:

create\_ll(&start);

printf("\n LINKED LIST CREATED");

break;

case 2:

display(start);

break;

case 3:

pop(&start);

break;

case 4:

end\_delete(&start);

break;

case 5:

delete\_at\_pos(&start);

break;

case 6:

free\_list(start);

printf("\nExiting....\n");

break;

}

} while (option != 6);

return 0;

}

void create\_ll(struct node \*\*start)

{

struct node \*new\_node, \*ptr;

int num;

printf("Enter -1 to end\n");

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

scanf("%d", &num);

while (num != -1)

{

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

if (new\_node == NULL)

{

printf("Memory allocation failed\n");

exit(EXIT\_FAILURE);

}

new\_node->data = num;

new\_node->next = NULL;

if (\*start == NULL)

{

\*start = new\_node;

}

else

{

ptr = \*start;

while (ptr->next != NULL)

ptr = ptr->next;

ptr->next = new\_node;

}

printf("\nEnter the data : ");

scanf("%d", &num);

}

}

void display(struct node \*start)

{

struct node \*ptr = start;

while (ptr != NULL)

{

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

ptr = ptr->next;

}

}

void pop(struct node \*\*start)

{

if (\*start == NULL)

{

printf("List is empty\n");

return;

}

struct node \*ptr = \*start;

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

free(ptr);

}

void end\_delete(struct node \*\*start)

{

if (\*start == NULL)

{

printf("List is empty\n");

return;

}

struct node \*ptr = \*start;

struct node \*ptr1 = NULL;

while (ptr->next != NULL)

{

ptr1 = ptr;

ptr = ptr->next;

}

if (ptr1 != NULL)

{

ptr1->next = NULL;

free(ptr);

}

else

{

// Only one node in the list

free(ptr);

\*start = NULL;

}

}

void delete\_at\_pos(struct node \*\*start)

{

if (\*start == NULL)

{

printf("List is empty\n");

return;

}

int loc;

printf("\nEnter the location of the node which has to be deleted : ");

scanf("%d", &loc);

struct node \*ptr = \*start;

struct node \*ptr1 = NULL;

for (int i = 0; i < loc; i++)

{

ptr1 = ptr;

ptr = ptr->next;

if (ptr == NULL)

{

printf("There are less than %d elements in the list\n", loc);

return;

}

}

if (ptr1 != NULL)

{

ptr1->next = ptr->next;

free(ptr);

printf("Deleted node at %d position\n", loc);

}

else

{

// Deleting the first node

\*start = ptr->next;

free(ptr);

printf("Deleted node at %d position\n", loc);

}

}

void free\_list(struct node \*start)

{

struct node \*ptr = start;

struct node \*next\_node;

while (ptr != NULL)

{

next\_node = ptr->next;

free(ptr);

ptr = next\_node;

}

}



