11/10/23

Deletion linked list week 4

```
#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 ** \n");
            printf ("\n 1 create a list ");
            printf ("\n 2 Display the list ")
            printf( "\n 3 Delete a node from the
                    beginning ");

            printf ("\n4 Delete a node from the
                    end ");
            printf( "\n 5 Delete 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:  pp (&start);
             break;

    case 4:  end_delete (&start);
             break;

    case 5:  delete_at_pos (&start)
             break;

    case 6:  free_list (start);
             printf("\n Exiting...\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");
    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 ("\n Enter 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"),
        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
    {
        free (ptr)
        *start = NULL;
    }
}

void delete_at_pos (struct node ** start)
{
    if (*start == NULL)
    {
        printf ("list is empty\n"),
        return;
    }
int loc

printf(" \n enter 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 list \n", loc);
          return;
     }
}
if (ptr1 != NULL)
{
     ptr -> next = ptr -> next
     free (ptr)
     printf(" Deleted node at %d position \n", loc);
}
else
{
     *start = ptr -> next;
     free (ptr);
     printf (" delete 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;
    }
}
```

→ Output

```
Main Menu
1. Create a SLL
2. Display the list
3. Delete node from beginning
4. Delete node from end
5. Delete from specific position
6. Exit
Enter your option : 1
Enter -1 to end
Enter data : 10
Enter data : 20
Enter data : 30
Enter data : 40
Enter data : -1
Enter your option : 2

        10          20          30          40


Enter your option : 3

Enter your option:

                20          30          40


Enter your option : 4

                20          30


Enter your option : 5
Enter the location of node to be deleted : 1


Enter your option : 2

                20
```

18/1/24