Singly Linked List Implementation in C

```c
#include <stdio.h>

#include <stdlib.h>


// Define the node structure

struct Node {

    int data;

    struct Node *next;

};


// Function to create a new node

struct Node* createNode(int data) {

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

    if (!newNode) {

        printf("Memory allocation failed.\n");

        exit(1);

    }

    newNode->data = data;

    newNode->next = NULL;

    return newNode;

}


// Function to insert a node at the beginning

void insertAtBeginning(struct Node **head, int data) {

    struct Node *newNode = createNode(data);
    newNode->next = *head;

    *head = newNode;

    printf("Node with data %d inserted at the beginning.\n", data);

}



// Function to delete a node with a specific value

void deleteNode(struct Node **head, int key) {

    struct Node *temp = *head, *prev = NULL;



    // If the head node itself holds the key

    if (temp != NULL && temp->data == key) {

        *head = temp->next;

        free(temp);

        printf("Node with data %d deleted.\n", key);

        return;

    }



    // Search for the key in the list

    while (temp != NULL && temp->data != key) {

        prev = temp;

        temp = temp->next;

    }



    // If the key was not present

    if (temp == NULL) {

        printf("Node with data %d not found.\n", key);

        return;
    }

    // Unlink the node from the list
    prev->next = temp->next;
    free(temp);
    printf("Node with data %d deleted.\n", key);
}

// Function to traverse the linked list
void traverseList(struct Node *head) {
    if (head == NULL) {
        printf("The linked list is empty.\n");
        return;
    }
    printf("Linked list contents: ");
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

// Main function to demonstrate linked list operations
int main() {
    struct Node *head = NULL;

    // Insert nodes into the list
```

```c
    insertAtBeginning(&head, 10);

    insertAtBeginning(&head, 20);

    insertAtBeginning(&head, 30);



    // Traverse the list

    traverseList(head);



    // Delete a node

    deleteNode(&head, 20);

    traverseList(head);



    // Attempt to delete a node that doesn't exist

    deleteNode(&head, 40);



    return 0;

}
```