Отчёт по лабораторной №2. Списки 2.

Задание. Вариант 1. Написать программу, позволяющую редактировать строку в процессе ввода: добавлять символы, удалять, перемещать и т.д. Использовать двусвязный список.

Реализация на языке C.

```c
#include <stdio.h>
#include <stdlib.h>

// Узел двусвязного списка
typedef struct Node {
    char data;
    struct Node *prev;
    struct Node *next;
} Node;

typedef struct {
    Node *head;
    Node *tail;
    Node *cursor;
} DoublyLinkedList;

// Функция создания нового узла
Node* createNode(char data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Функция инициализации списка
DoublyLinkedList* createList() {
    DoublyLinkedList* list = (DoublyLinkedList*)malloc(sizeof(DoublyLinkedList));
    list->head = NULL;
    list->tail = NULL;
    list->cursor = NULL;
    return list;
}

// Вставка символа перед курсором
void insertBeforeCursor(DoublyLinkedList* list, char data) {
    Node* newNode = createNode(data);
    if (list->cursor == NULL) {
        if (list->head == NULL) {
            list->head = newNode;
            list->tail = newNode;
        } else {
            newNode->next = list->head;
            list->head->prev = newNode;
            list->head = newNode;
        }
        list->cursor = newNode;
    } else {
        newNode->next = list->cursor;
        newNode->prev = list->cursor->prev;
        if (list->cursor->prev) {
```

```
            list->cursor->prev->next = newNode;
        } else {
            list->head = newNode;
        }
        list->cursor->prev = newNode;
        list->cursor = newNode;
    }
}

// Удаление символа под курсором
void deleteAtCursor(DoublyLinkedList* list) {
    if (list->cursor == NULL) return;
    Node* temp = list->cursor;

    if (temp->prev) {
        temp->prev->next = temp->next;
    } else {
        list->head = temp->next;
    }
    if (temp->next) {
        temp->next->prev = temp->prev;
    } else {
        list->tail = temp->prev;
    }
    list->cursor = temp->next ? temp->next : temp->prev;
    free(temp);
}

// Перемещение курсора влево
void moveCursorLeft(DoublyLinkedList* list) {
    if (list->cursor && list->cursor->prev) {
        list->cursor = list->cursor->prev;
    }
}

// Перемещение курсора вправо
void moveCursorRight(DoublyLinkedList* list) {
    if (list->cursor && list->cursor->next) {
        list->cursor = list->cursor->next;
    }
}

// Вывод текущего состояния строки
void printList(DoublyLinkedList* list) {
    Node* current = list->head;
    while (current) {
        if (current == list->cursor) {
            printf("|");  // Курсор
        }
        printf("%c", current->data);
        current = current->next;
    }
    if (list->cursor == NULL) {
        printf("|");  // Курсор
    }
    printf("\n");
}

// Очистка памяти
void freeList(DoublyLinkedList* list) {
    Node* current = list->head;
```

```
    while (current) {
        Node* next = current->next;
        free(current);
        current = next;
    }
    free(list);
}

int main() {
    DoublyLinkedList* list = createList();
    char command;

    printf("Редактор строки\n");
    printf("Команды:\n");
    printf("i <символ> - вставить символ перед курсором\n");
    printf("d - удалить символ под курсором\n");
    printf("l - переместить курсор влево\n");
    printf("r - переместить курсор вправо\n");
    printf("q - выйти из редактора\n");

    while (1) {
        printf("Текущая строка: ");
        printList(list);
        printf("Введите команду: ");
        scanf(" %c", &command);

        switch (command) {
            case 'i': {
                char ch;
                scanf(" %c", &ch);
                insertBeforeCursor(list, ch);
                break;
            }
            case 'd':
                deleteAtCursor(list);
                break;
            case 'l':
                moveCursorLeft(list);
                break;
            case 'r':
                moveCursorRight(list);
                break;
            case 'q':
                freeList(list);
                return 0;
            default:
                printf("Неизвестная команда\n");
                break;
        }
    }

    return 0;
}
```

Пример работы программы.

Взаимодействие с программой: