ЛАБОРАТОРНА РОБОТА № 6

Лінійні однозв'язані і двозв'язані списки

Мета роботи: отримати навички в організації і обробці однозв'язаних і двозв'язаних списків, а також навчитися їх використовувати при вирішенні завдань.

Короткі теоретичні відомості

Зв'язний список – базова структура даних, в якій кожен елемент містить інформацію, необхідну для отримання наступного елемента. Основна перевага зв'язаних списків перед масивами полягає в можливості ефективного змінювати їх вигляд. За цю гнучкість доводиться жертвувати швидкістю доступу до довільного елементу списку, оскільки єдиний спосіб отримання елемента полягає у відстеженні зв'язків від початку списку.

Зв'язний список – це набір елементів, причому кожний з них є частиною вузла (*node*), який також містить посилання (*link*) на вузол. Вузли визначаються посиланнями на вузли, тому зв'язані списки іноді називають самопосилальними (*self-referent*) структурами. Більш того, хоча вузол зазвичай посилається на інший вузол, можливе посилання на самого себе, тому зв'язані списки можуть являти собою циклічні (*cyclic*) структури (рис. 6.1).

Рис. 6.1. Приклад взаємозв'язку елементів списків

Щоб задати динамічний список треба описати його вузол. Оскільки вузол складається з полів різних типів, то описати його можна неоднорідним типом – структурою.

Завдання типу елемента списку, який зберігає дані цілого типу

```
typedef struct Node {
    int item;
    Node * next;
} Node;
```

Щоб працювати зі списком як з єдиним об'єктом, треба ввести статичну змінну-покажчик, значення якої – це адреса першого (або заголовного) елемента списку. Якщо список порожній, вона повинна мати значення *NULL*.

Основними операціями обробки списку є:

1) пошук заданого елемента за його значенням або порядковим номером; операція завершується, коли елемент знайдений або переглянутий весь список (якщо елемента немає); результат операції повинен визначати, чи є елемент в списку чи ні і, якщо є, то можливо повернути його адресу або значення;

2) включення (вставка) в список нового елемента перед або після заданого елемента (в тому числі перед першим елементом або після останнього). Включенню, як правило, передує пошук елемента, після і/або перед яким відбувається включення; при включенні елемента перед першим в список змінюється заголовок списку; при включенні після деякого елемента змінюється поле посилання, після якого відбувається включення, тому треба визначати посилання на елемент, після якого відбувається включення (рис. 6.2);

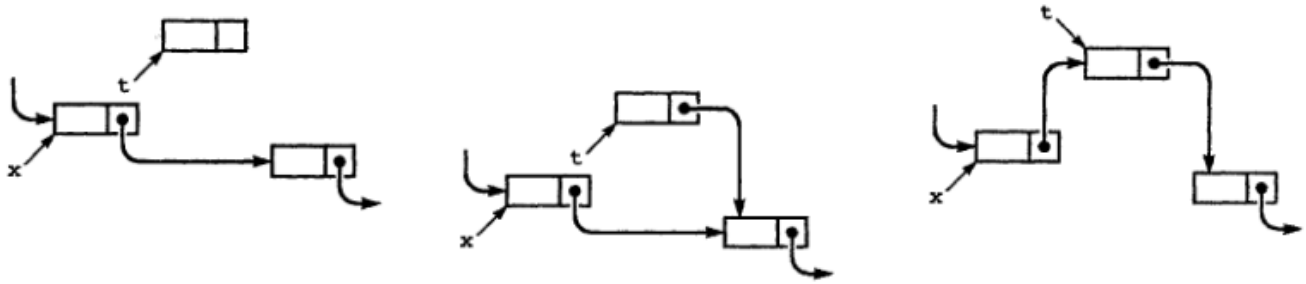

Рис. 6.2. Включення елементів списку

3) виключення (видалення) заданого елемента зі списку (в тому числі видалення першого елемента або останнього). Виключенню, як правило, передує пошук елемента, що виключається; результатом пошуку повинне бути посилання на елемент, що передує елементу, який виключається зі списку, так як при видаленні елемента зі списку змінюється поле посилання для елемента, що передує видаленому; при видаленні першого елемента змінюється заголовок списку (рис. 6.3);

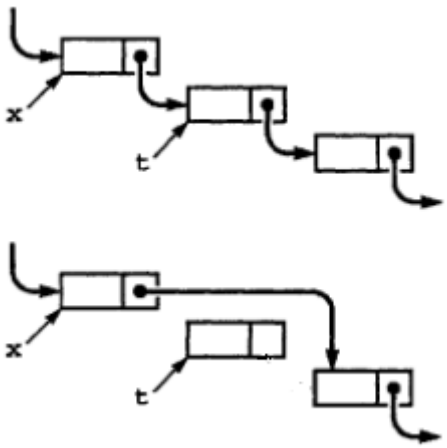

Рис. 6.3. Виключення заданого елемента зі списку

4) визначення числа елементів списку;

5) впорядкування елементів списку за значенням інформаційного поля.

Перевага зв'язаних списків перед масивами полягає в тому, що зв'язані списки ефективно змінюють розміри протягом життя. Зокрема, необов'язково заздалегідь знати максимальний розмір списку. Одне з важливих практичних наслідків цієї властивості полягає в можливості мати кілька структур даних, що мають спільний простір, не приділяючи особливої уваги їх відносному розміру в будь-який момент часу.

Реалізація простих операцій вставки і видалення елемента в списку:

```
// додавання нового елемента в початок списку
void push(Node **head, int data) {
    Node *tmp = (Node*) malloc(sizeof(Node));
    tmp-> item = data;
    tmp-> next = (*head);
    (*head) = tmp;

// видалення першого елемента в списку
int pop(Node **head) {
    Node* prev = NULL;
    int val;
    if (head == NULL) {
        exit(-1);
    }
    prev = (*head);
    val = prev-> item;
    (*head) = (*head)->next;
    free(prev);
    return val;
}
```

Реалізація функції друку елементів списку

```
void printLinkedList(const Node *head) {
    for (Node *t = head; t!= NULL; t = t->next) {
        printf("%d ", t->item);
    }
    printf("\n");
}
```

Динамічний список, в якому кожен елемент (крім, можливо, першого і останнього) пов'язаний з попереднім і наступним елементами, називається **двозв'язаним**. Кожен елемент такого списку має два поля з посиланнями: одне поле містить посилання на наступний елемент, інше поле – посилання на попередній елемент і третє поле – інформаційне. Наявність посилань на наступний і попередній вузли дозволяє рухатися по списку від кожного вузла в будь-якому напрямку: від вузла до кінця списку або від вузла до початку списку, поэтому такой список называют еще и двунаправленным (рис. 6.4).

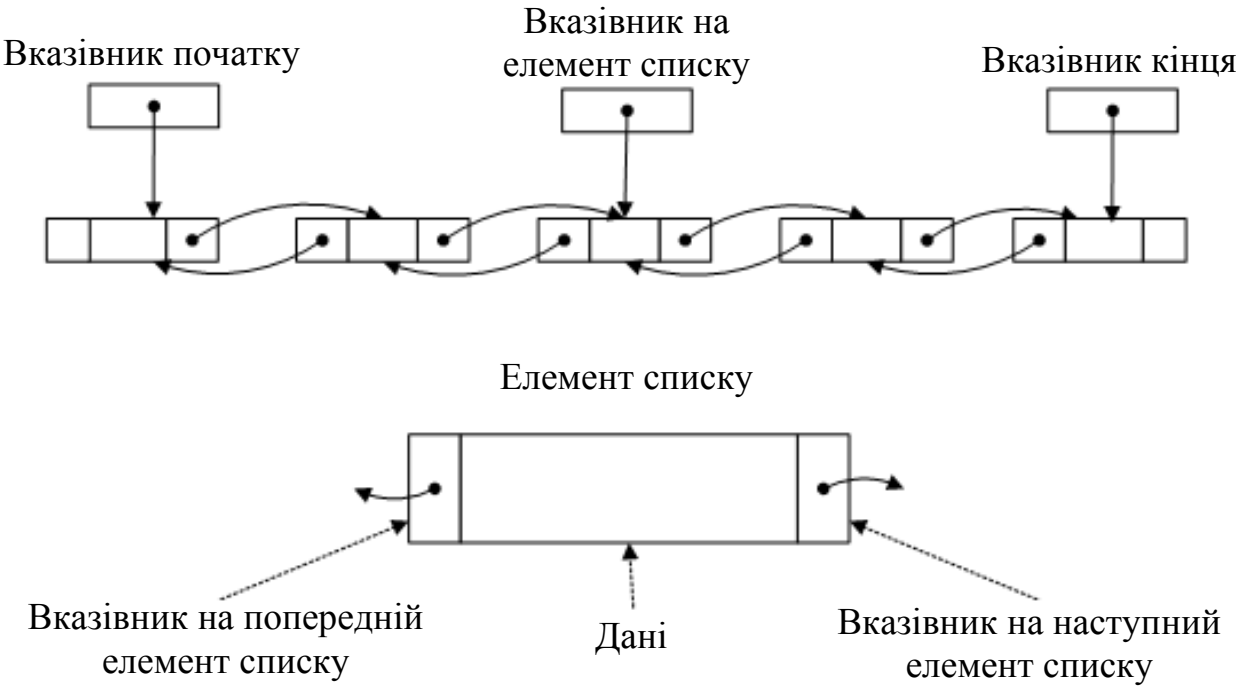

Рис. 6.4. Взаємодія елементів двонапрямлених списків

Завдання типу елемента двозв'язаного списку, який зберігає дані цілого типу:

```
typedef struct NodeT {
int item;
Node * next, * prev;
} NodeT;
```

Основні операції, що виконуються над двозв'язаним списком, ті ж, що і для однозв'язного списку. Оскільки двозв'язаний список більш гнучкий, ніж однозв'язний, то при включенні елемента в список, можна використовувати вказівник як на елемент, за яким відбувається включення, так і вказівник на елемент перед яким відбувається включення. При виключенні елемента зі списку можна використовувати як вказівник на сам елемент, що виключається, так і вказівник на елемент, що йому передує або наступний за ним. Але оскільки елемент двозв'язаного списку має два вказівника, то при виконанні операцій включення / виключення елемента треба змінювати більше зв'язків, ніж для однозв'язного списку. Пошук елемента двозв'язаного списку можна здійснювати:

а) переглядаючи елементи від початку до кінця списку,

б) переглядаючи елементи від кінця списку до початку,

в) переглядаючи список в обох напрямках одночасно: від початку до середини списку і від кінця до середини (враховуючи, що елементів в списку може бути парне або непарна кількість).

Завдання до лабораторної роботи № 6.

Для **кожного варіанта** – реалізувати однозв'язний і двозв'язний список і операції роботи з ними. Непарні варіанти зберігають в списку цілі значення, парні – символи. Списки заповнити з клавіатури. Операції:

- додати на початок;
- додати в кінець;
- додати в середину (після зазначеного за значенням елемента);
- видалити (з будь-якого місця списку);
- знайти за значенням;
- роздрукувати список.

Використовуючи два створені списки, виконати два завдання за варіантом. Варіант визначається номером студента у списку групи. Варіанти завдань визначати згідно з таблицею:

Варіант	Завдання	Варіант	Завдання	Варіант	Завдання
1	1, 13	**11**	1, 11	**21**	7, 13
2	2, 14	**12**	2, 12	**22**	8, 14
3	3, 11	**13**	3, 13	**23**	3, 5
4	4, 12	**14**	4, 14	**24**	6, 10
5	5, 9	**15**	1, 7	**25**	3, 7
6	6, 10	**16**	2, 6	**26**	4, 6
7	5, 7	**17**	3, 9	**27**	9, 13
8	6, 8	**18**	4, 8	**28**	10, 12
9	7, 9	**19**	5, 11		
10	8, 10	**20**	6, 12		

Завдання:

1. Створити новий (третій) однозв'язний список. Помістити в нього всі парні елементи з перших двох. Результуючий список вивести на екран.
2. Створити новий (третій) однозв'язний список. Помістити в нього перші три елементи з першого і другого списків. Результуючий список вивести на екран.
3. Створити новий (третій) однозв'язний список. Помістити в нього всі непарні елементи з перших двох. Результуючий список вивести на екран.
4. Створити новий (третій) однозв'язний список. Помістити в нього перші п'ять елементів з першого і другого списків. Результуючий список вивести на екран.
5. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, які більше заданого числа. Результуючий список вивести на екран.
6. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, які більше заданого символу. Результуючий список вивести на екран.

7. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, які менше заданого числа. Результуючий список вивести на екран.

8. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, які менше заданого символу. Результуючий список вивести на екран.

9. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, за винятком нульового елемента. Результуючий список вивести на екран.

10. Створити новий (третій) однозв'язний список. Помістити в нього всі елементи з перших двох, за винятком символу *'a'*. Результуючий список вивести на екран.

11. Створити новий (третій) однозв'язний список. Помістити в нього всі додатні елементи з перших двох. Результуючий список вивести на екран.

12. Створити новий (третій) однозв'язний список. Помістити в нього всі символи нижнього регістра з перших двох. Результуючий список вивести на екран.

13. Створити новий (третій) однозв'язний список. Помістити в нього всі від’ємні елементи з перших двох. Результуючий список вивести на екран.

14. Створити новий (третій) однозв'язний список. Помістити в нього всі символи верхнього регістру з перших двох. Результуючий список вивести на екран.