**Контрольні питання**

1. Існують задачі, для яких:

складно (або неможливо) до початку роботи програми визначити обсяг даних, які необхідно обробити;

в процесі роботи необхідно досить часто використовувати операції вставки/видалення, перестановки елементів.

Для таких задач використовуються динамічні структури даних.

1. ДСД розділяються на:
   1. лінійні (списки, стек, черга);
   2. ієрархічні (дерева, графи).
2. **struct** node {

**int** value;

node \*next;

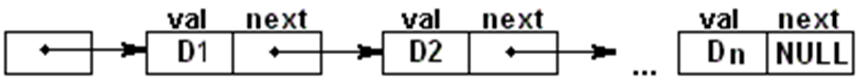
node ( **int** x ) {

value = x;

next = NULL;

}

};



1. Опис, побудова, руйнування, виведення елементів на екран, пошук елемента, видалення елемента, вставка ел..
2. Отже, ми можемо додавати елементи в початок списку і в його кінець. Що нам це дає? У тому вигляді, в якому він реалізований зараз, немає особливих переваг перед звичайним однозв’язним списком. Але якщо додати гетери для полів head і tail, користувач нашого списку зможе реалізувати множину різних алгоритмів. Крім того, двозв’язний список дозволяє легко реалізувати двозв’язну чергу, яка є будівельним блоком для інших структур даних.
3. Стек – LIFO

Стек може бути реалізований як динамічний список, для якого операції додавання елемента та видалення елемента здійснюється тільки з кінця списку.

1. Черга – FIFO

Черга може бути реалізована як динамічний список, для якого операції додавання елемента здійснюється тільки з кінця списку, видалення елемента – тільки з початку списку.