Лабораторна робота №4

«Зв’язний список, стек, черга. Зворотній польський запис»

Мета роботи: ознайомитися з основами роботи з двозв’язним списком, однозв’язним списком, стеком та чергою. Розробити основні функції для обчислення арифметичного виразу, записаного з використанням зворотного польського запису.

Методичні вказівки

Зв’язний список (linked list) — це структура даних, у якій об’єкти розташовані у лінійному порядку. Однак, на відміну від масиву, у якому цей порядок визначається індексами, порядок у зв’язному списку визначається вказівниками на кожен об’єкт.

Зв’язний список є динамічною структурою, що дає змогу записувати нові елементи в будь-яке місце вже існуючого списку і так само читати елементи в будь-якому місці.

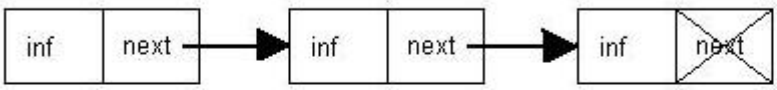
Зв’язні списки забезпечують просте й гнучке представлення динамічних множин і підтримують вище перераховані операції.

Зв’язний список — це різновид лінійних структур даних, що є послідовністю елементів, зазвичай відсортованих відповідно до деякого критерію. Послідовність може містити будь-яку кількість елементів, оскільки при створенні списку використовується динамічний розподіл пам’яті.

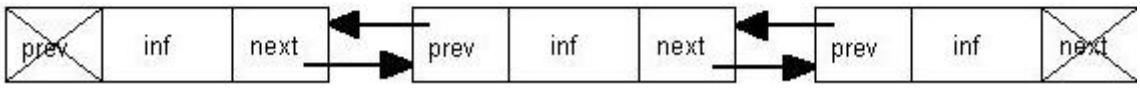
Кожний елемент зв’язного списку є окремим об’єктом (або структурою), що містить поля для зберігання інформації та вказівник на наступний елемент списку. (Якщо список є двозв’язним, то в такому об’єкті зберігається також вказівник на попередній елемент.)

Схематично одно- та двозв’язний список можна представити наступним чином.

Связный список



Двусвязный список

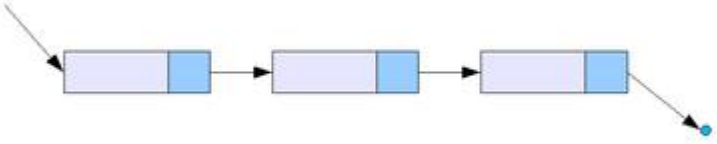


Пересування по списку здійснюється за вказівниками, які містять адресу сусіднього елемента списку. При додаванні до списку нового елемента необхідно динамічно виділити для нього пам’ять за допомогою функції malloc (оператора new) та присвоїти відповідні адреси вказівникам сусідніх елементів.

Види зв’язних списків

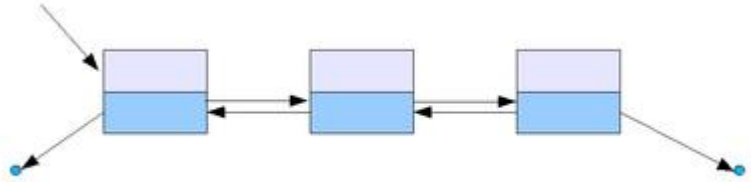
Існує декілька можливих типів зв’язних списків.

Однозв’язний список



В однозв’язному списку можна пересуватися лише у напрямку від першого до останнього вузла. При цьому довідатися адресу попереднього елемента неможливо.

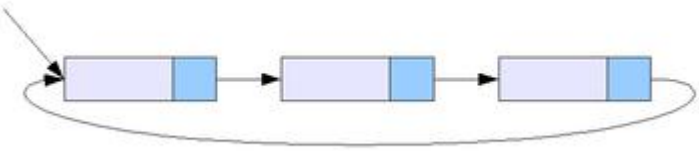
Двозв’язний список



По двозв’язному списку можна пересуватися в будь-якому напрямку — як від першого елементу у напрямку останнього, так і у зворотному напрямку. У цьому списку простіше робити видалення та перестановку елементів, оскільки завжди відомі адреси попереднього та наступного елементів списку.

Кільцевий зв’язний список

Різновидом зв’язних списків є кільцевий (або циклічний, замкнутий) список. Він теж може бути одно- або двозв’язним. Останній елемент кільцевого списку містить покажчик на перший елемент, а перший (для двозв’язного списку) — на останній.



Черга працює за принципом першим прийшов – першим пішов.

Стек працює за принципом останнім прийшов – першим пішов.

Стек і черга є також лінійними списками, множина допустимих операцій над якими обмежена операціями над першим або останнім елементом.

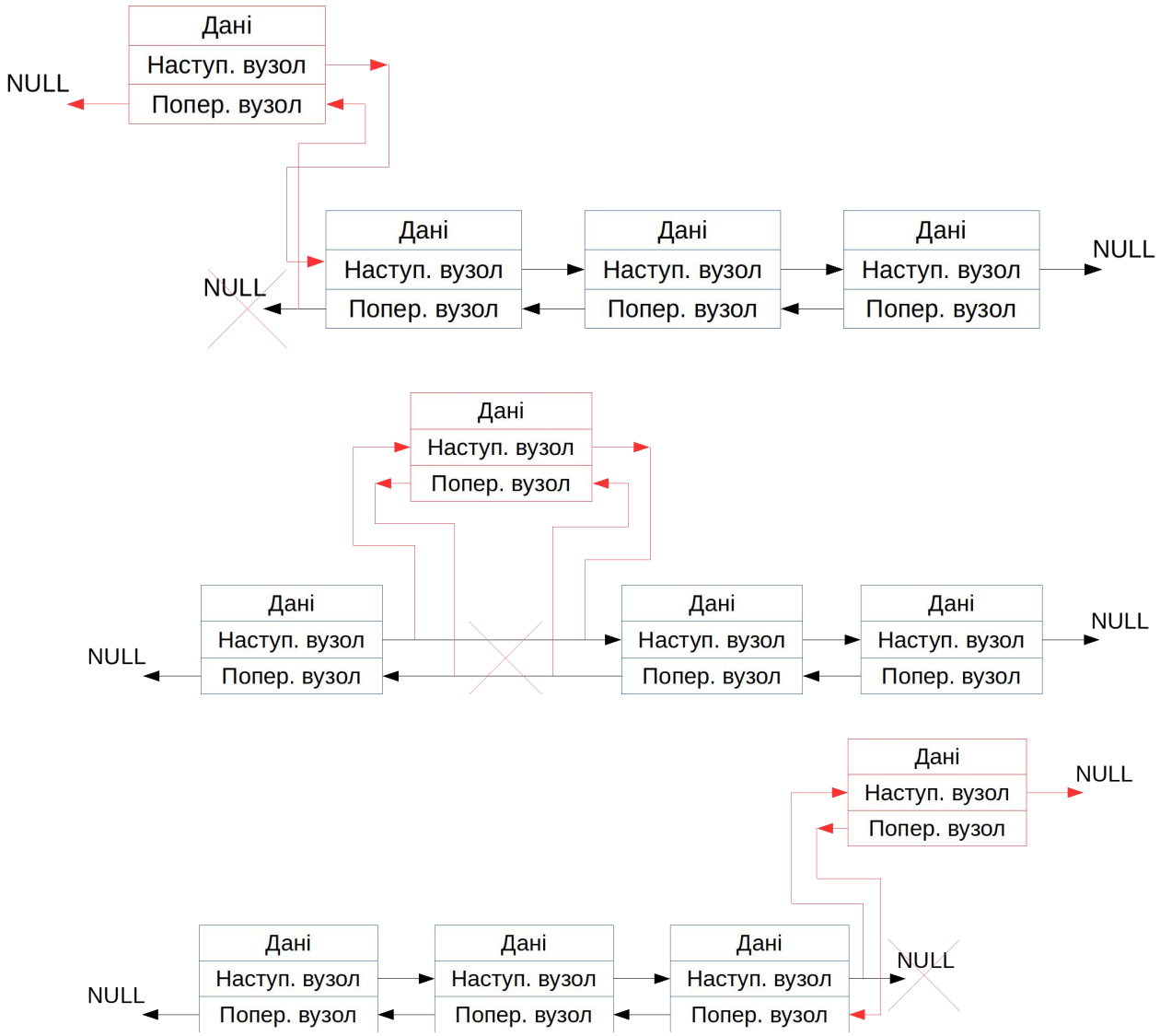
Найбільш ефективно у спискових структурах реалізуються операції вставки та видалення елементів, оскільки вони, на відміну від операцій видалення та вставки елементів масиву, не потребують зсуву групи елементів. Наведемо всі можливі варіанти застосування цих двох операцій:

- додавання елемента на початої списку;
- додавання елемента в середину списку;
- додавання елемента в кінець списку;
- видалення елемента з початку списку;
- видалення елемента з середини списку;
- видалення елемента з кінця списку.

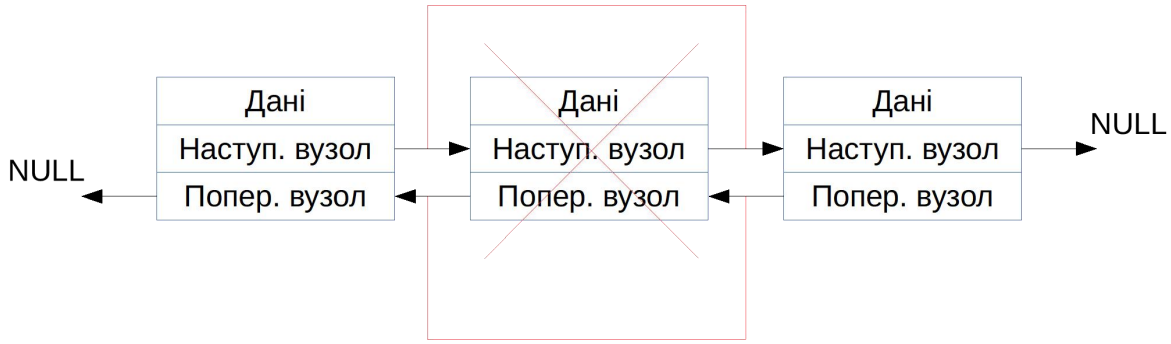
У загальному випадку для роботи з двозв'язним лінійним списком потрібні такі покажчики: покажчик head на початок списку; покажчик current на поточний елемент списку; покажчики prev та next на елемент, розташований перед поточним та після; та покажчик tail на кінець списку. Зауважимо, що у розв'язаннях конкретних задач можуть використовуватися не всі такі покажчики.

Отже, згадаємо деякі алгоритми основних операцій над двозв'язним лінійним списком.

Алгоритм вставки елементів списку



Алгоритм видалення елемента зсередини списку



Додавання елемента в кінець списку виконується за алгоритмом додавання елемента до черги, а додавання елемента на початок списку - за алгоритмом додавання елемента до стеку.

В математиці оператор розміщують між операндами (x + y), а не після операндів (xy +). Форма з оператором між операндами називається **інфіксним записом**. Форма з оператором після операндів називається постфіксним або зворотнім польським записом (ЗПЗ) у честь польського логіка Я. Лукасевича, який вивчав властивості цього запису.

Наприклад вираз (7 2 3 * -) еквівалентно виразу в інфіксной нотації: (7 - 2 * 3).

Зворотній польський запис має ряд переваг перед інфіксним записом при вираженні алгебраїчних формул:

- формула може бути виражена без дужок;

- запис зручна для обчислення формул в машинах зі стеками;

- інфіксні оператори мають пріоритети, які довільні і небажані. Наприклад, ми знаємо, що **ab + c** значить **(ab) + c**, а не **a (b + c)**.

До недоліків польського запису можна віднести неможливість використання однакових символів для операцій, наприклад унарний та бінарний мінус.

Для перетворення виразу з інфіксной форми у зворотню польску запис Е. Дейкстра винайшов алгоритм. Алгоритм отримав назву «Сортувальна станція» за схожість його операцій з тим, що відбувається на залізничних сортувальних станціях.

Алгоритм має наступний вигляд:

1.	Поки є ще символи для читання:					
	1.	Читаємо черговий символ.				
	2.	Якщо символ є числом або постфіксною функцією (наприклад,! - факторіал), додаємо його до вихідного рядку.				
	3.	Якщо символ є префіксною функцією (наприклад, sin - синус), поміщаємо його у стек.				
	4.	Якщо символ є відкриваючою дужкою, поміщаємо його у стек.				
	5.	Якщо символ є закриваючою дужкою:				
		1.	Поки верхнім елементом стека не стане відкриваюча дужка:			
			1.	Виштовхуємо елементи зі стеку у вихідний рядок.		
			інакше:			
			1.	Відкриваюча дужка видаляється зі стеку, але у вихідний рядок не додається.		
	6.	Якщо символ є бінарною операцією o1, тоді:				
		1.	Поки на вершині стека префіксна функція			
			або операція на вершині стека більш пріоритетна за o1			
			або операція на вершині стека лівоасоціативна з пріоритетом як у o1			
			1.	Виштовхуємо верхній елемент стека у вихідний рядок.		
2.	Коли вхідний рядок закінчився, виштовхуємо все символи зі стеку у вихідний рядок.					

Розглянемо цей алгоритм на прикладі виразу:

A+B*(C/B+Z*(A+D))

Символ	**Стек**	**Строка**	**Символ**	**Стек**	**Строка**
A		A	*	+*(+*	ABCB/Z
+	+	A	(	+*(+*(	ABCB/Z
B	+	AB	A	+*(+*(	ABCB/ZA

*	+*	AB	+	+*(+*(+	ABCB/ZA
(	+*(	AB	D	+*(+*(+	ABCB/ZAD
C	+*(	ABC	)	+*(+*	ABCB/ZAD+
/	+*(/	ABC	)		ABCB/ZAD+*+*+
B	+*(/	ABCB			
+	+*(+	ABCB/			
Z	+*(+	ABCB/Z			

Алгоритм обчислення виразу, записаного в ЗПЗ наступний (для реалізації цього алгоритму використовується стек):

1. Якщо на вхід подано операнд, то він поміщається у вершину стека, а якщо на вхід поданий знак операції, то відповідна операція виконується над необхідною кількістю значень, витягнутих зі стека, взятих в порядку додавання. Результат виконаної операції вставляється назад у вершину стека;

2. Якщо весь вхідний набір символів оброблений, то переходимо до пункту 1;

3. Коли весь вхідний набір буде оброблений, то в стеці залишиться одне значення, яке буде результатом цього виразу.

Розглянемо цей алгоритм на прикладі виразу:

7 5 2 - 4 * +

Символ	**Дія**	**Стан стеку після скоєної дії**
7	Покласти у стек	7
5	Покласти у стек	7 5
2	Покласти у стек	7 5 2
-	Витягуємо зі стеку 2 верхніх числа і здійснюємо операцію "-". Результат поміщаємо у стек	7 3
4	Покласти у стек	7 3 4
*	Витягуємо зі стеку 2 верхніх числа і здійснюємо операцію "*". Результат поміщаємо у стек	7 12
+	Витягуємо зі стеку 2 верхніх числа і здійснюємо операцію "+". Результат поміщаємо у стек	19

Порядок виконання роботи

1. Розробити бібліотеку всіх основних функцій роботи з двозв’язним списком. Створити програму тестування бібліотеки роботи з двозв’язним списком. Створення та заповнення динамічних структур даних повинно виконуватися в діалоговому режимі. Програма повинна виконувати наступні операції: створення списку, додавання елементів (всі 3), видалення елементів (всі 3), виведення списку на дисплей, знищення списку. Протестуйте програму для 7 – 10 елементів для звіту.

Формат вхідних даних:

Декілька рядків команд. Кожен рядок починається з команди, а через пробіл параметр, якщо він є. Кожна команда це операція зі списком, вивід даних, вихід з програми та

має свій код. Ось перелік всіх команд:

0 - вихід,

1- очищення списку,

2- додавання елемента на початок списку (вхідний параметр - значення, яке додається),

3- додавання елемента в кінець списку (вхідний параметр - значення, яке додається),

4- додавання елемента у середину списку (два вхідних параметри: 1. позиція в яку додається значення, 2. саме значення, яке додається),

5- видалення елемента на початку списку,

6- видалення елемента з середини списку (вхідний параметри: - позиція елемента в списку, яка видаляється)

7- видалення елемента в кінці списку,

8- виведення списку на дисплей.

Формат вихідних даних:

Перелік цілих чисел через пробіл, які знаходяться у списку.

Приклад:

Послідовність введення команд:

1
2 5
2 3
3 7
8
0

Результат виведення списку у консоль.

3 5 7

2. Розробити програму обчислення арифметичного виразу, скориставшись алгоритмом польського запису. Операнди у виразі розділяти пробілами. Операції: додавання (+), віднімання (-), множення (*), ділення (/), зведення в ступінь (^), корінь числа (реалізувати через зведення у ступінь).

Формат вхідних даних:

Математичний вираз у інфіксній формі (розділяти пробілами).

Формат вихідних даних:

Результат обчислення.

Приклад:

Послідовність введення виразу:

(2 + 3) / 5

На наступному рядку результат обчислення виразу:

1

Контрольні запитання

1. Дайте означення поняттю зв’язний список.
2. Які види зв’язних списків Ви знаєте?
3. Чим відрізняється принцип дії зв’язного списку від принципу роботи з елементами черги та стеку?
4. Поясніть, яким чином описується список.
5. Поясніть процедуру перегляду елементів черги.

6. Поясніть процедуру перегляду елементів стеку.