# ЛЕКЦІЯ 7. ЛІНІЙНІ СТРУКТУРИ ДАНИХ.

# СТЕКИ, ЧЕРГИ І ДЕКИ.

**Стеки, черги і деки** - це однотипні лінійні структури. Їх називають динамічними лінійними структурами даних у зв'язку з тим, що вибірка елемента із цих послідовностей зводиться до операції виключення, тобто ліквідації його у послідовностях. Відрізняються ці структури одна від одної порядком виконання операцій включення і виключення елементів .

До основних операцій над цими структурами відносять [2]:

а) створення нової структури даних;

б) запис або включення елемента у вже існуючу структуру даних;

в) виключення елемента зі структури;

г) перевірка умови існування структури.

## 7.1. Стеки

**Стек** - це впорядкована лінійна динамічно змінювана послідовність елементів, в якій виконуються наступні умови:

1) новий елемент приєднується завжди до одного і того ж боку послідовності;

2) доступ до елемента здійснюється завжди з того боку послідовності, до якого приєднується елемент;

3) елемент зберігається в послідовності до моменту його виклику .

Прикладом стеку є стопка тарілок, а також магазин патронів у рушниці. За аналогією з останнім прикладом стек ще часто називають магазинною пам'яттю.

Операцію включення елемента у стекназивають "проштовхуванням", а операцію виключення із послідовності - "виштовхуванням". Найбільш і найменш доступні елементи стека навивають відповідно верхом і низом стека. Операції включення і виключення зі стека виконують в одному і тому ж боку послідовності, але виключаються елементи зі стека в порядку, зворотному до того, в якому вони потрапили в послідовність. В кожний момент часу зістека можна забрати лише один елемент. Дисципліну обслуговування стека часто називають дисципліною ***LIFO*** - "останній прийшов, перший пішов".

Стек можна зобразити у вигляді послідовності $A=A_1,...,A_{n-1},A_n$ , де $A_n$ -елемент, що вказує на вхід-вихід у послідовність; $A_1$ , $A_n$ - відповідно низ і верх стека. Щоб прочитати елемент $A_j$ , необхідно вибрати зі стека елементи $A_n,..., A_{j+1}$.

Також стек можна представити у вигляді трубки з підпруженим дном, що розміщена вертикально. Верхній кінець трубки відкритий, у нього можна включати, або заштовхувати елементи. Новий елемент, що додається, проштовхує елементи, включені в стек раніше, на одну позицію вниз. При виключенні елементів зі стека вони виштовхуються нагору.

Незалежно від того, як реалізований стек, процедури, що працюють із ним, повинні вміти поміщати елементи в стек, вибирати елементи зі стеку й перевіряти, не чи порожній стек. Для того щоб можна було працювати з новим типом даних, необхідні спеціальні операції, наприклад:

NEW (стек) створення порожнього стека

PUSH (стек, елемент даних) розміщення елемента в стек

POP (стек) вибірка елемента з вершини стека

EMPTY (стік) перевірка, чи порожній стек; результатом є значення "істина", якщо стек порожній, і "хибно " в протилежному випадку.

NEW і PUSH - базові операції, використовувані при роботі зі стеком. Перша створює стек, а друга поміщає в нього елементи. POP - процедура, характерна тільки для роботи зі стеком. Вона може бути визначено за допомогою двох базових проміжних операцій TOP і REMOVE, які дозволяють виділити дві основні функції, реалізовані POP, але не визначають способів організації доступу до структури з інших модулів:

Відповідно наведеним вище визначенням, POP вибирає вершину стека й видаляє її зістека. Вершина стека описується в такий спосіб: якщо стек не порожній, вершина являє собою останній розміщений у стекелемент; а якщо ні, то стек не має вершини. Операція REMOVE дає в результаті стек без елемента, розміщеного в нього останнім. До тільки що створеногостеку операцію REMOVE застосовувати не можна. Для визначення порожньогостека використовується поняття нового (або тільки що створеного) стека: порожній стек не містить жодного елемента.

Будь-яка реалізація стека повинна задовольняти цим визначенням. Важливо, що стек визначається як спосіб зберігання даних, які підпорядковуються певним правилам, що діють при звертанні до даних, а не як масив з вказівниками, які пересуваються за певними правилами.

Основною перевагою стека перед іншими організаціями даних є те, що у ньому не потрібна адресація елементів. Для обслуговування стека потрібно лише дві команди: PUSH - "проштовхнути" і POP -"виштовхнути". Зі стеком пов'язаний завжди один вказівник, який вказує на верхній елемент у стеку. На початку такий вказівник дорівнює нулю.

Найпростішим прикладом вдалого застосування стека може служити задача перепису вектора у зворотному порядну. На практиці стекова структура даних найчастіше застосовується в рекурсивнихалгоритмах, при трансляції, а також при обробці переривань програм.

### 7.1.1. Реалізація стеку на базі масиву

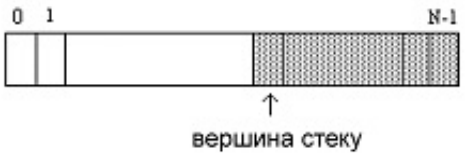
Реалізація стеку на базі масиву є класикою програмування. Іноді навіть саме поняттястеку не цілком коректно ототожнюється із цією реалізацією.

Базою реалізації є масив розміру *N*. Таким чином реалізується стек обмеженого розміру, максимальна глибина якого не може перевищувати *N*. Індекси елементівмасиву змінюються від 0 до *N - 1*. Елементи стеку зберігаються в масиві в такий спосіб: елемент на дні стеку розташовується на початку масиву, тобто в елементі з індексом *0*. Елемент, розташований над самим нижнім елементом стеку, зберігається в елементі з індексом *1*, і так далі. Вершина стеку зберігається десь у середині масиву. Індекс елемента на вершині стеку зберігається в спеціальній змінній, яку звичайно називають **вказівником** стеку (англійською Stack Pointer або просто SP).


Коли стек порожній, вказівник стеку містить значення мінус одиниця. При включенні елемента вказівник стеку спочатку збільшується на одиницю, потім у елемент масиву з індексом, що зберігається в вказівнику стеку, записується елемент, що включається. При видаленні елемента

зі стеку спершу вміст елемента масиву з індексом, що зберігається у вказівнику стеку, запам'ятовується в тимчасовій змінній у якості результату операції, потім вказівникстеку зменшується на одиницю.

У наведеній реалізації стек росте убік збільшення індексів елементів масиву. Часто використовується інший варіант реалізації стеку на базі вектора, коли дностеку міститься в останньому елементімасиву, тобто в елементі з індексом *N - 1*. Елементи стеку займають безперервний відрізок масиву, починаючи із елемента, індекс якого зберігається у вказівникустеку, і закінчуючи останнім елементоммасиву. У цьому варіанті стек росте убік зменшення індексів. Якщо стек порожній, то вказівник стеку містить значення *N* , яке на одиницю більше, ніж індекс останнього елемента масиву).




## 7.2. Черги

**Черга** - це лінійна динамічно змінювана послідовність елементів, у якій виконуються такі умови:

1) новий елемент приєднується завжди з одного і того ж боку послідовності;

2) доступ до елементів або їх виключення завжди здійснюється з другого боку послідовності.

Наприклад, нехай послідовність $Q=Q_1... ,Q_n$ зображує чергу. Тоді елемент $Q_1$, називають "головою" черги і він вказує на місце для виключення елемента; а елемент $Q_n$ називають "хвостом" черги і він вказує на місце для включення елемента в чергу.

Чергу можна представити у вигляді трубки. В один кінець трубки можна додавати кульки — елементи черги, з іншого кінця вони видаляються. Елементи в середині черги, тобто кульки усередині трубки, недоступні. Кінець трубки, у який додаються кульки, відповідає кінцю черги, кінець, з якого вони видаляються — початку черги. Таким чином, кінці трубки не симетричні, кульки усередині трубки рухаються тільки в одному напрямку.

Отже, в структурі черг потрібні два вказівники: один - для посилання на вершину послідовності (для включення елемента), другий - для посилання на основу послідовності (для виключення елемента). При кожному виключенні елемента із такої структури виключається завжди найстаріший елемент. Черги обслуговуються за дисципліною ***FIFO*** -"перший прийшов, перший пішов".

Черги бувають різних типів: лінійні, з пріоритетом і циклічні. Звичайна **лінійна**черга може бути зображена масивом, з двома вказівниками: перший вказує на елемент для вибірки з черги, другий - на останній елемент, записаний у чергу. Багаторазове звертання до елементів лінійної черги призводить до того, що пам'ять для зберігання такої черги використовується неефективно. У лінійній черзі після вибірки елемента пам'ять, зайнята чергою, звільняється і повторно не використовується, оскільки нові елементи приписуються з іншого краю послідовності.

Чергу, для якої є можливість включати або виключати елементи з певної позиції в залежності від деяких її характеристик називають **пріоритетною** чергою. Прикладом пріоритетної черги може служити порядок розв'язування потоку задач на комп'ютері у деяких операційних системах . Така черга зводиться до послідовності лінійних черг, якщо відомі пріоритети її елементів. Кожна черга з послідовності обслуговується за дисципліною *FIFO* , але елементи з другої черги обслуговуються тільки тоді, коли порожня попередня черга, а з третьої тоді, коли порожні перша і друга черги. При включенні елементи приєднуються до боку однієї з черг згідно з їх пріоритетом.

Черги, в яких елементи $Q_1 ,Q_2, ...,Q_n$розміщуються так, що за елементом $Q_n$ іде

елемент $Q_1$ називаються **циклічними**(рис. 7.1). Циклічну чергу також можна зобразити лінійною послідовністю, але при записі нового елемента на відміну від лінійної черги вся послідовність зсувається на одне поле і новий елемент записується знову на її початку. Вибірка елемента буде також виконуватися за вказівником на кінець послідовності.

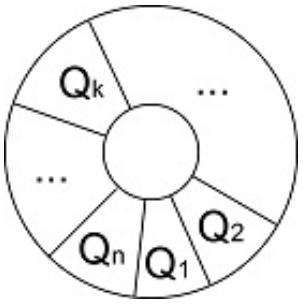



*Рис.7.1. Графічне зображення циклічної черги*

Прикладом циклічної черги може служити робота обчислювальної системи з розподілом часу, з якою одночасно працює багато користувачів. Оскільки така система має переважно один блок обробки, що називають процесором, і одну пам′ять, то в кожний момент часу ці ресурси належать одному користувачу. Для кожного користувача виділяється певний інтервал часу. Тільки на відміну від пріоритетної черги одна задача до кінця не розв′язується, а "порціями" протягом виділеного їй інтервалу часу. Програми, що чекають на виконання, утворюють циклічну чергу.

### 7.2.1. Використання черги в програмуванні

Використання черги в програмуванні майже відповідає її ролі у звичайному житті. Черга практично завжди пов'язана з обслуговуванням запитів, у тих випадках, коли вони не можуть бути виконані миттєво. Черга підтримує також порядок обслуговування запитів.

Будь-яка, навіть найпростіша, операційна система завжди тією чи іншою мірою багатозадачна. Це значить, що в момент натискання клавіші операційна система може бути зайнята якою-небудь іншою роботою. Проте, операційна система ні в якій ситуації не має права проігнорувати натискання на клавішу. Тому відбувається переривання роботи комп'ютера, він запам'ятовує свій стан і перемикається на обробку натискання на клавішу. Така обробка повинна бути дуже короткою, щоб не порушити виконання інших завдань. Команда, що віддається натисканням на клавішу, просто додається в кінець черги запитів, що чекають свого виконання. Після цього переривання закінчується, комп'ютер відновлює свій стан і продовжує роботу, яка була перервана натисканням на клавішу. Запит, поставлений у чергу, буде виконаний не відразу, а тільки коли настане його черга.

У системі Windows робота віконних додатків заснована на повідомленнях, які посилають цим додаткам. Наприклад, бувають повідомлення про натискання на клавішу миші, про закриття вікна, про необхідність перемальовування області вікна, про вибір пункту меню й т.п. Кожна програма має чергу запитів. Коли програма одержує свій квант часу на виконання, вона вибирає черговий запит з початку черги й виконує його. Таким чином, робота віконного додатка полягає в послідовнім виконанні запитів з її черги.Черга підтримується операційною системою.

Підхід до програмування, що полягає не в прямому виклику процедур, а в посиланні повідомлень, які ставляться в чергу запитів, має багато переваг і є однією з рис об'єктно-орієнтованого програмування. Так, наприклад, якщо віконній програмі необхідно завершити роботу з якої-небудь причини, краще не викликати відразу команду завершення, яка небезпечна, тому що порушує логіку роботи й може привести до втрати даних. Замість цього програма посилає самій собі повідомлення про необхідність завершення роботи, яке буде поставлено в чергу запитів і виконане після запитів, що зробили раніше.

### 7.2.2. Реалізація черги на базі масиву

Усі структури даних можна реалізовувати на основі масиву. Звичайно, така реалізація може бути багатоетапною, і не завжди масив виступає як безпосередня база реалізації. У випадку черги найбільш популярні дві реалізації: безперервна на базі масиву, яку називають також реалізацією на базі кільцевого буфера, і посилальна реалізація, або реалізація на базі списку.

При безперервній реалізації черги в якості бази виступає масив фіксованої довжини*N*, таким чином, черга обмежена й не може містити більш *N* елементів. Індекси елементів масиву змінюються в межах від*0* до *N - 1*. Крім масиву, реалізація черги зберігає три прості змінні: індекс початку черги, індекс кінця черги, число елементів черги. Елементи черги зберігаються у відрізку масиву від індексу початку до індексу кінця.

При додаванні нового елемента в кінець черги індекс кінця спочатку збільшується на одиницю, потім новий елемент записується в елемент масиву із цим індексом. Аналогічно, при видаленні елементу з початку черги вміст елементумасиву з індексом початку черги запам'ятовується як результат операції, потім індекс початку черги збільшується на одиницю. Як індекс початку черги, так і індекс кінця при роботі рухаються зліва направо. Що відбувається, коли індекс кінця черги досягає кінця масиву, тобто *N - 1*?

Ключова ідея реалізації черги полягає в тому, що масив подумки як би зациклюється в кільце. Вважається, що за останнім елементом масиву розміщений його перший елемент (останній елемент має індекс *N-1*, а перший - індекс *0*). При зрушенні індексу кінця черги вправо у випадку, коли він указує на останній елемент масиву, він переходить на перший елемент. Таким чином, безперервний відрізок масиву, зайнятий елементами черги, може переходити через кінець масиву на його початок.

Циклічну чергу найкраще зображати циклічним або кільцевим списком, доповненим вказівниками на позицію для включення і виключення елементів із черги.

## 7.3. Деки

**Деки** - це впорядкована лінійна динамічно змінювана послідовність елементів, в якій виконуються такі умови:

1) новий елемент може приєднуватися з обох боків послідовності;

2) вибірка елементів можлива також з обох боків послідовності.

Наприклад, якщо послідовність $D = D_1, \ldots, D_n$ зображує дек, то її елементи $D_1$ і $D_n$ вказують одночасно на місце включення і виключення елементів.

Деки бувають з обмеженнями з одного боку на виконання однієї з операцій (наприклад, один бік послідовності може бути закритий для вибірки елементів і відкритий тільки для приєднування, і навпаки).

Деки називають також **реверсивними**чергами або чергами з двома кінцями, в яких включення і виключення здійснюються з двох країв послідовності. Як і у випадку звичайних черг, для обслуговування деків також потрібно два вказівники, по одному на кожний кінець послідовності. Деки є найбільш загальною лінійною динамічною структурою даних.