**14. РЕКУРСИВНІ СТРУКТУРИ ДАНИХ**

**10.1.** Описати клас для реалізації стеку на базі зв’язування об’єктів. Передбачити виконання дій над стеком:

1) Почати роботу.

2) Чи порожній стек?

3) Вштовхнути елемент у стек.

4) Верхівка стеку.

5) Забрати верхівку стеку.

Використовуючи цей клас, розв’язати задачу: на вхід поступає послідовність. Впорядкувати цю послідовність за зростанням.

*Вказівка*. Для впорядкування використати 2 стеки.

**10.2.** Використовуючи клас для реалізації стеку на базі зв’язування об’єктів (див. завдання 10.1), скласти підпрограми:

а) довжина стеку (запобігти знищеннню стеку під час обчислення його довжини);

б) інверсiя стеку;

**10.3.** Описати клас для реалізації черги на базі зв’язування об’єктів. Передбачити виконання дій над чергою:

1) Почати роботу.

2) Чи порожня черга?

3) Додати елемент до кінця черги.

4) Взяти елемент з початку черги.

Використовуючи цей клас, скласти підпрограму обчислення довжини черги. Запобігти знищенню черги після виклику підпрограми. Передбачити рекурсивний та нерекурсивний варіант.

**10.4.** Використовуючи клас для реалізації черги (див. завдання 10.3), розв’язати наступну задачу. У черзі є *m* чисел. Проводять *n* випробувань, в результаті кожного з яких отримують випадкові числа 0 або 1. Якщо отримано 0, то треба взяти елемент з початку черги та показати його на екрані. Якщо отримано 1, то ввести число з клавіатури та додати до кінця черги. Після завершення випробувань показати залишок черги.

*Вказівка*. Використати генератор випадкових чисел.

**10.6.** Використовуючи клас для реалізації черги (див. завдання 10.3), розв’язати наступну задачу. У магазині стоїть черга з *m* покупців. Час обслуговування покупця з черги – це випадкове ціле число в діапазоні від 1 до *t1*. Час додавання нового покупця до черги - це випадкове ціле число в діапазоні від 1 до *t2*. Промоделювати стан черги (тобто показати час виникнення подій – обслуговування та додавання покупця) за період часу *T* (*T* >> *t1*, *T* >> *t2*). Показати залишок черги.

**10.7.** Використовуючи клас для реалізації черги (див. завдання 10.3), скласти підпрограми:

а) інверсiя черги;

б) забрати *n* елементiв черги;

в) конкатенацiя двох черг;

г) порiвняти 2 черги.

**10.8.** Описати клас для реалізації деку на базі списку. Передбачити виконання дій над деком:

1) Почати роботу.

2) Чи порожній дек?

3) Додати елемент до початку деку.

4) Взяти елемент з початку деку.

5) Додати елемент до кінця деку.

6) Взяти елемент з кінця деку.

Використовуючи цей клас, скласти підпрограми: обчислення довжини деку, копіювання деків. Запобігти знищенню деку після виклику підпрограми обчислення його довжини.

**10.9.** Використовуючи клас для реалізації деку (див. завдання 10.8), реалізувати стек на базі деку. Реалізувати стек на базі деку - означає описати клас та реалізувати дії над стеком викликами відповідних підпрограм, що реалізують дії над деком. Для реалізованого стеку розв’язати задачу інвертування вхідної послідовності.

**10.10.** Використовуючи клас для реалізації деку (див. завдання 10.8), реалізувати чергу на базі деку (див. попереднє завдання). Для реалізованої черги розв’язати задачу обчислення довжини черги.

**10.12.** Використовуючи клас для реалізації деку (див. завдання 10.8), розв’язати задачу 10.4, передбачивши однак чотири можливих результати кожного випробування (випадкові числа від 0 до 3):

 0 – взяти елемент з початку деку та показати його на екрані;

 1 – ввести число з клавіатури та додати його до початку деку;

 2 – взяти елемент з кінця деку та показати його на екрані;

 3 – ввести число з клавіатури та додати його до кінця деку.

**10.13.** Використовуючи клас для реалізації деку (див. завдання 10.8), розв’язати задачу 10.6, передбачивши що через випадковий час від 1 до *t3* до початку черги додається „пільговий” покупець, який обслуговується першим, а через випадковий час від 1 до *t4* не витримує та йде з черги останній покупець.

**10.14.** Використовуючи клас для реалізації деку (див. завдання 10.8), скласти підпрограми:

а) інверсiя деку;

б) конкатенацiя двох декiв;

в) порiвняти 2 деки;

г) забрати *n* елементiв з початку деку;

д) забрати *n* елементiв з кiнця деку;

**10.20.** Описати клас для реалізації кільцевого списку на базі зв’язування об’єктів. Передбачити виконання дій над списком:

1) Почати роботу.

2) Довжина списку.

3) Перейти до наступного елемента.

4) Поточний елемент.

5) Вставити елемент.

6) Видалити елемент.

Використовуючи цей клас, розв’язати задачі:

а) “лічилка” ;

б) пошук у списку елемента, рівного заданому числу *Search*(*L*, *x*) та присвоєння для списків *Let*(*L1*, *L2*);

в) знайти послідовність рівних елементів списку, що йдуть підряд, максимальної довжини;

г) видалити із списку всі повторні входження елементів;

д) чкщо, список складається з чисел, знайти пару елементів списку, різниця між якими є максимальною за абсолютною величиною для всіх пар елементів списку.

**10.21.** Використовуючи клас для реалізації кільцевого списку цілих чисел (див. завдання 10.20), скласти підпрограми:

а) *Change*(*L*, *n*) - замiнити поточний елемент списку *L* числом *n*;

б) *Copy*(*L*, *m*, *n*, *L1*) - видiлити з списку *L* *n* елементiв, починаючи з елемента з номером *т* у новий список *L1*;

в) *Del*(*L*, *m*, *n*) - видалення *n* елеменiв списку *L*, починаючи з *m*-го, по відношенню до поточного, елемента кільцевого списку.

**10.25.** Описати клас для реалізації бінарного дерева на базі списку. Передбачити виконання дій над деревом:

1) Почати роботу.

2) Чи порожнє дерево?

3) Створити дерево.

4) Корінь дерева.

5) Лівий син.

6) Правий син.

7) Змінити корінь

8) Змінити правого сина

9) Змінити лівого сина

Використовуючи цей модуль, розв’язати задачі:

а) виведення дерева на екран *Print*(*t*);

б) пошуку у дереві елемента, рівного заданому *Search*(*t*, *x*);

в) побудови бінарного дерева пошуку та пошуку елемента у ньому (бінарне дерева називають деревом пошуку, якщо для будь-якого його піддерева корінь цього піддерева не менше кожної вершини лівого сина та не більше кожної вершини правого сина);

г) обчислення висоти дерева *Height*(*t*);

д) перевірки, чи входить одне дерево у другие *IsIn*(*t1*, *t2*).

**10.26.** Описати клас для реалізації сильно розгалуженого дерева. Вершину дерева представити у вигляді кортежу (<навантаженя>, <список синів>). Передбачити виконання дій над деревом:

1) Почати роботу.

2) Чи порожнє дерево?

3) Створити дерево з заданим навантаженням та списком синів.

4) Повернути корінь дерева.

5) Повернути список синів.

6) Змінити корінь дерева

7) Змінити список синів

Використовуючи цей клас, розв’язати задачі:

а) побудови бінарного дерева за сильно розгалуженим деревом;

б) пошуку у дереві елемента, рівного заданому числу *Search*(*t*, *x*);

в) обчислення висоти дерева *Height*(*t*);

г) перевірки, чи входить одне дерево у другие *IsIn*(*t1*, *t2*).

**10.27.** Описати клас для реалізації орієнтованих графів на базі зв’язування об’єктів. Передбачити виконання дій над графом:

1). Створити порожній граф

2). Вершини графа

3). Довжина графа

4). Повернути вершину

5). Повернути дані вершини

6). Повернути список попередників

7). Повернути список наступників

8). Оновити дані вершини

9). Оновити список попередників

10). Оновити список наступників

11). Видалити вершину

12). Оновити (додати) вершину

Використовуючи цей клас, розв’язати задачі:

а) перевірити, чи існує шлях між двома вершинами;

б) знайти найкоротший шлях між двома вершинами.

в) знайти найдовший шлях, що не є циклом та діаметр графу (довжина цього шляху)

г) перевірити, чи є граф сильно зв’язним (граф є сильно зв’язним, якщо між будь-якими двома вершинами існує шлях).

Т14.1 Побудувати підпрограми, що розв’язують задачі для орієнтованих графів:

а) перевірити чи містить граф цикли

б) знайти довжину найдовшого циклу

в) побудувати список усіх дуг графу (дугу представити кортежем з двох вершин)

г) побудувати список ізольованих вершин графу (з напівстепінню входу та виходу 0)