15. Обробка помилок та виключних ситуацій

Т15.1. Скласти підпрограму та програму для обчислення значення натурального числа за заданим рядком символів, який є записом цього числа у системі числення за основою b($2<b<16$). Використати функцію, яка за заданим символом повертає відповідну цифру у системі числення за основою b. Використати у цій функції твердження про стан програми assert для перевірки того, що відповідний символ є цифрою у системі числення за основою b. Обробити у підпрограмі помилку неправильного символу рядка та показати змістовне повідомлення про помилку.

Т15.2. Скласти функцію та програму для обчислення наближеного значення суми

$$y = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots (|x| < 1);$$

Використати у цій функції твердження про стан програми assert для перевірки того, що параметр x відповідає заданій умові. Обробити у програмі помилку неправильного значення x та показати змістовне повідомлення про помилку.

Т15.3. Скласти функцію та програму для обчислення суми всіх доданків, модуль яких не менше $\varepsilon > 0$, у комплексній точці z

$$arctg\ z = z - \frac{z^3}{3} + \frac{z^5}{5} - \ldots + (-1)^n \frac{z^{2n+1}}{2n+1} + \ldots \quad (|z| < 1);$$

Використати у цій функції твердження про стан програми assert для перевірки того, що параметр z відповідає заданій умові. Обробити у програмі помилку неправильного значення z та показати змістовне повідомлення про помилку.

Т15.4. Скласти функцію зі змінною кількістю параметрів та програму для обчислення

$$f(x_1, \ldots, x_n, y_1, \ldots, y_n) = \sum_{i=1}^{n} x_i^2 + y_i^2 + x_i \cdot y_i$$

Вказівка: оформити x_i як позиційні, а y_i, - як ключові параметри.
Використати у цій функції твердження про стан програми assert для перевірки того, що кількість параметрів x_i дорівнює кількості параметрів y_i. Обробити у програмі помилку неправильної кількості параметрів та показати змістовне повідомлення про помилку.

Т15.5. Задані натуральне число *i* файл *f*, компоненти якого є цілими числами. Побудувати файл *g*, записавши в нього найбільше значення перших *n* компонент файлу *f*, потім-наступних *n* компонент і т.д. Розглянути два випадки:
а) число компонент файлу ділиться на *n*;
б) число компонент файлу не ділиться на *n*.
В цьому випадку остання компонента файлу *g* повинна дорівнювати найбільшій із компонент файлу *f*, які утворюють останню (неповну) групу.
Забезпечити обробку помилок при роботі з файлами.

Т15.6. У текстовому файлі записана непорожня послідовність дійсних чисел, які розділяються пропусками в одному рядку та можуть бути розташовані у різних рядках. Визначити функцію обчислення найбільшого з цих чисел.
Забезпечити обробку помилок, якщо у файлі зустрічаються не дійсні числа.

Т15.7. Описати клас Двохбайтне ціле число для роботи з цілими числами, представленими двома байтами. Інтервал представлення при цьому – від -2^{15} (-32768) до 2^{15}-1 (32767). Операції не можуть вивести за межі інтервалу представлення. Наприклад, 32767 + 1 == -32768, 32767 + 2 == - 32767 і т.д. Якщо результат операції виводить за межі інтервалу представлення, повинна ініціюватися помилка переповнення.
Перевизначити у цьому класі операції +, -, *, //, %.
Описати також 3 класи обробки помилок для двохбайтних цілих чисел: загальний клас обробки помилок та два його підкласи для обробки помилки переповнення та помилки ділення на 0.
Використати цей клас для розв’язання задач:
а) обчислення *n*!
б) обчислення x^n, де *x* – ціле, *n* – невід’ємне ціле.
Забезпечити обробку помилок при виконанні обчислень.

Т15.8. Описати клас Поліном та реалізувати методи: введення поліному, виведення поліному, обчислення значення поліному у точці x, взяття похідної поліному, суми, різниці та добутку поліномів.
Описати також клас обробки помилок при неправильному введенні поліному (степінь – не невід’ємне ціле число, коефіцієнт – не дійсне число) та забезпечити ініціювання помилки при неправильному введенні.
Використати цей клас для розв’язання задачі: ввести 2 поліноми P1, P2 та рядок, який містить вираз, що залежить від 2 поліномів. Наприклад,
P1 + P2*P1 – P2
Обчислити поліном, який буде значенням цього виразу.
Забезпечити обробку помилок неправильного введення поліному.

Вказівка: поліном представити у вигляді словника.

Т15.9. Описати клас для реалізації мультимножини на базі словника. Мультимножина - це множина в якій для кожного елемента запам'ятовується не лише його входження, але й кількість входжень.

Кількість входжень елемента *k* (0<=*k*<=*n*) у мультимножину - це значення елемента словника з ключем *k*.

Реалізувати дії над мультимножинами:

1) зробити мультимножину порожньою;
2) чи є мультимножина порожньою;
3) додати елемент до мультимножини;
4) забрати елемент з мультимножини (кількість входжень елемента зменшується на 1, якщо елемент не входить - відмова);
5) кількість входжень елемента у мультимножину;
6) об'єднання двох мультимножин (в результаті об'єднання кількість входжень елемента визначається як максимальна з двох мультимножин);
7) перетин двох мультимножин (в результаті кількість входжень елемента визначається як мінімальна з двох мультимножин);

Описати клас обробки помилки взяття елементу, який не входить до мультимножини.

З використанням класу розв'язати задачі:

а) перевірити, чи складаються рядки *S1*, *S2* з одних і тих же символів, які входять у ці рядки однакову кількість разів;

б) перевірити, чи вірно, що всі символи рядка *S1*, входять також у рядок *S2*, причому не меншу кількість разів, ніж у *S1*.

Забезпечити обробку помилок.

Т15.10. Описати клас для реалізації стеку та клас обробки помилки взяття елементу порожнього стеку та отримання верхівки порожнього стеку. Передбачити виконання дій над стеком:

1) Почати роботу.
2) Чи порожній стек?
3) Вштовхнути елемент у стек.
4) Верхівка стеку.
5) Забрати верхівку стеку.

Передбачити ініціювання виключення у разі взяття елементу порожнього стеку та отримання верхівки порожнього стеку.

Використовуючи цей клас, розв'язати задачу:

У стеку є *n* чисел. Проводять *m* випробувань, в результаті кожного з яких отримують випадкові числа 0 або 1. Якщо отримано 0, то треба отримати та показати верхівку стеку а також забрати верхівку стеку. Якщо отримано 1, то ввести число з клавіатури та вштовхнути у стек. Після завершення випробувань показати залишок стеку.

Забезпечити обробку помилки взяття елемента з порожнього стеку під час проведення випробувань.

Вказівка. Використати генератор випадкових чисел.

Т15.11. Описати клас для реалізації деку. 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.

Описати клас обробки помилок взяття елемента з початку та з кінця порожнього деку.
Забезпечити ініціювання помилки у разі взяття елемента з початку та з кінця порожнього деку.
Використовуючи цей клас, розв'язати задачу Т15.10, 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.

Забезпечити обробку помилки взяття елемента з порожнього деку під час проведення випробувань.

Т15.12. Використовуючи клас для реалізації деку та клас обробки помилок деку (див. завдання Т15.11), розв'язати задачу:
У магазині стоїть черга з m покупців. Час обслуговування покупця з черги – це випадкове ціле число в діапазоні від 1 до t_1. Час додавання нового покупця до черги - це випадкове ціле число в діапазоні від 1 до t_2. Через випадковий час від 1 до t_3 до початку черги додається „пільговий" покупець, який обслуговується першим, а через випадковий час від 1 до t_4 не витримує та йде з черги останній покупець. Промоделювати стан черги (тобто показати час виникнення подій – обслуговування та додавання покупця) за період часу T ($T >> t_1$, $T >> t_2$, $T >> t_3$, $T >> t_4$). Показати залишок черги покупців.
Передбачити обробку помилки взяття елементу з порожнього деку (при обслуговуванні покупця, якщо черга порожня).

Т15.13. Використовуючи клас для реалізації деку та клас обробки помилок деку (див. завдання Т15.11), скласти підпрограми:
а) забрати n елементів з початку деку;
б) забрати n елементів з кінця деку;
Передбачити обробку помилки взяття елементу з порожнього деку.

Т15.14. Описати клас для реалізації кільцевого списку та клас обробки помилок отримання поточного елементу та видалення елементу порожнього списку. 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, скласти підпрограми:
а) *Change*(*L*, *n*) - замінити поточний елемент списку *L* числом *n*;
б) *Copy*(*L*, *m*, *n*, *L1*) - виділити з списку *L n* елементів, починаючи з елемента з номером *m* у новий список *L1*;
в) *Del*(*L*, *m*, *n*) - видалення *n* елементів списку *L*, починаючи з *m*-го, по відношенню до поточного, елемента кільцевого списку.
Передбачити обробку помилок отримання поточного елементу та видалення елементу порожнього списку.

Т15.15. Описати клас для реалізації орієнтованих графів. 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

Описати 4 класи, що утворюють ієрархію класів обробки помилок у графах: батьківський клас (помилка у графі), помилка отримання/зміни навантаження (або списку попередників, списку наступників) для неіснуючої вершини, неправильні параметри підпрограми «Оновити (додати) вершину», невалідне посилання на неіснуючу вершину при додаванні/оновленні вершини.

Використовуючи ці класи, 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).

Забезпечити обробку помилок під час дій над графами.