ІНДИВІДУАЛЬНІ ЗАВДАНЯ для лабораторної роботи № 2
з дисципліни «Програмування-ІІ»
("Об'єктно-орієнтоване програмування")

кафедра комп'ютерних технологій, ДНУ
2018/2019 н.р.

Тема: "Використання класів в мові С++"

Постановка задачі

Розробити об'єктно-орієнтовану бібліотеку для роботи зі структурами даних за однією з нижченаведених тем у відповідності з нижченаведеними вимогами. Властивості та методи для класів розробити у відповідності з відомими визначеннями відповідних структур даних. Скласти тести для перевірки працездатності бібліотеки. Скласти програму, що демонструє можливості розробленої бібліотеки.

Загальні вимоги

В незалежності від індивідуального варіанта повинні бути реалізовані наступні можливості:

1. Реалізація методів ініціалізації (конструктор по замовчуванню та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []), присвоювання (перевантаження =), візуалізації, збереження (на диск) та відновлення, діалогового керування, "розумного доступу" (перевантаження ->), а також псевдо змінних (забезпечення можливості виду: f(x)=const).
2. Перевантаження (спільне використання) потокового введення/виведення. (введення з файл, виведення в файл)
3. Створення та використання файла бібліотеки (*.LIB).
4. Повторне використання класів без їх перекомпіляції (ReUse).
5. Застосування вказаної структури даних для розв'язання типової задачі.

Результати виконання лабораторної роботи повинні бути викладені у вигляді звіту. Загальні вимоги до звіту – у файлі (lab_treb.doc)

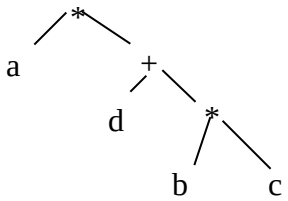
В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому режимі (за допомогою меню) виконувати операції над відповідними структурами даних.

Теми індивідуальних завдань

1. Стеки та обчислення виразів. Реалізувати розбір уведеного рядка для обчислення виразу. Вираз представляється в постфіксному вигляді, тобто $(2+3)\cdot 10$ представляється у вигляді $23+10\cdot$. Розбір рядка виконується таким чином: числа заштовхуються в стек доки не зустрінеться символ операції, потім зі стеку виштовхуються два елемента, виконується операція, результат операції заштовхується в стек. Використання псевдозмінних для стеку може бути виконано таким чином: по-перше і-му елементу в стеці може бути присвоєно деяке значення, по-друге, наприклад, при
s(i,j)=s2; //де s, s2 – стеки

в стеку s починаючи з i-го символу j елементів замінюються на стек s2.

2. Відображення (map). Діалогове керування передбачає таке: визначення типу даних вихідної множини, визначення типу даних множини відображення (з деякого обмеженого (не менше 3) набору типів), додавання до множини значень відображення пари „дані – відображені дані”, візуалізація результатів відображення. Реалізація псевдозмінних для відображення: замінити якусь частину відображення іншим відображенням (M1(2,5) =M2, тобто у відображенні M1 замінити пари елементів прообраз-образ з 2 по 5 включно відображенням M2).
3. Черги та пріоритети. При додаванні елемента до черги визначається пріоритет його обслуговування. При видаленні елементу з черги обирається елемент з більшим пріоритетом. Використання псевдозмінних для черги аналогічно використанню псевдозмінних для стека (див. варіант 1).
4. Черги и досяжність, шлях в лабіринті. Для знаходження шляху в лабіринті реалізувати алгоритм пошуку в ширину. Використання псевдозмінних для черги аналогічно використанню псевдозмінних для стека (див. варіант 1). Діалогове керування передбачає створення лабіринту $m \times n$, завдання точки, з якої починається пошук в лабіринті та кінцевої точки, візуалізацію заданого лабіринту, а також кроків пошуку шляху в лабіринті (тобто візуалізація на кожному кроці черги, в якій зберігається шлях).
5. Списки та функціональне програмування. Мови функціонального програмування (Lisp, ML, Haskell та ін.) містять єдину структура даних – список і єдину алгоритмічну структуру – функцію, яка також має вигляд списку. Необхідно скласти програму, яка забезпечує роботу користувача зі списками у стилі функціонального програмування.
   Користувач вводить:
   а) (= x (1 2 3)). В результаті програма виконує наступні дії :у список x заносяться значення 1,2,3.
   б) (cdr x) – в результаті на екран виводиться хвіст списку
   г) (=y (append x (-2 -1)) – в результаті список y буде містити наступні значення (1 2 3 (-2 -1))
   Використання псевдозмінних для списку аналогічно використанню псевдозмінних для стека (див. варіант 1).
6. Списки та символьні обчислення. Для виконання символьних обчислень можуть бути використані дерева розбору, наприклад, вираз $a \cdot (d + b \cdot c)$ може бути представлений у вигляді



   або у вигляді списків (a * (d + (b c *))). Використання псевдозмінних для списку аналогічно використанню псевдозмінних для стека (див. варіант 1). Передбачити можливість перевірки закону дистрибутивності.
7. Дерева бінарні: пошук і сортування (забезпечити сортування даних, що зберігаються в дереві).). Використання псевдозмінних для дерева може бути виконано таким чином:

по-перше елементу (i, j) в дереві (i – номер рівня, j – номер елемента на j-му рівні) може бути присвоєно деяке значення, по-друге, наприклад, при
d(i,j)=d2; //де d, d2 – дерева
замінити в дереві d j-й елемент на i-му рівні на піддерево d2.

8. Дерева довільні та генеалогія. Використання псевдозмінних для дерева може бути виконано таким чином: по-перше елементу (i, j) в дереві (i – номер рівня, j – номер елемента на j-му рівні) може бути присвоєно деяке значення, по-друге, наприклад, при
d(i,j)=d2; //де d, d2 – дерева
замінити в дереві d j-й елемент на i-му рівні на піддерево d2.
9. Вектори: сортування (по зростанню/зменшенню) та інші перетворення (множення вектора на число, обчислення довжини вектора, множення векторів, перевірка векторів на колінеарність, визначення кута між векторами, обчислення кута між вектором та віссю). Використання псевдозмінних для вектора аналогічно використанню псевдозмінних для стека (див. варіант 1). Реалізація псевдозмінних можу бути такою: d(0)=2; d(1)=3, де d – екземпляр класу раціонального числа, тоді в першому випадку значення чисельника буде дорівнювати 2, в другому випадку – значення знаменника дорівнюватиме 3.
10. Матриці: сортування та інші перетворення (підвищення в степінь, множення матриць, множення на константу, додавання матриць). Використання псевдозмінних для матриці може бути виконано таким чином: елементу (i,j) матриці присвоюється деяке значення. Реалізація псевдозмінних: m(3,5)=9, де m – екземпляр класу багаточлен, тоді коефіцієнти з третьої по п’яту степінь заміняться на 9.
11. Раціональні числа: різні форми зовнішнього представлення, точні обчислення. Реалізувати клас для збереження раціонального числа (чисельник, знаменник). Виконати перевантаження арифметичних операцій і операцій приведення типів. Передбачити автоматичне перетворення раціонального числа в нескорочуваний дріб. Реалізація псевдозмінних можу бути такою: d(0)=2; d(1)=3, де d – екземпляр класу раціонального числа, тоді в першому випадку значення чисельника буде дорівнювати 2, в другому випадку – значення знаменника дорівнюватиме 3.
12. Багаточлени (додавання, множення, множення на константу, ділення, підвищення в степінь). Реалізація псевдозмінних: m(3,5)=9, де m – екземпляр класу багаточлен, тоді коефіцієнти з третьої по п’яту степінь заміняться на 9.
13. Множини (додавання елемента до множини, видалення елемента з множини, перевірка наявності елемента в множині, об’єднання множин, перетин множин).
14. Довга арифметика: реалізація основних арифметичних операцій.
15. Оперативна пам’ять: стратегії динамічного розподілення. Розробка власного менеджера пам’яті. Виділення пам’яті за принципом: перший блок, що підходить за розміром, найбільш підходящий (залишок мінімального розміру), найменш підходящий (залишок максимального розміру); реалізувати стратегію відновлення: якщо два вільних блока пам’яті знаходяться рядом, то вони об’єднуються в один блок. Діалогове керування: виділити пам’ять, звільнити пам’ять, візуалізувати стан пам’яті (наприклад у вигляді 00000011111000000000111, де 111 – зайняті блоки). Використання псевдозмінних для оперативної пам’яті може бути виконано таким чином:
o1(i,j)=o2;
де o1, o2 – екземпляри класу «оперативна пам’ять». Блоки o1 замінюються на блоки o2, починаючи з i по j елемент;
o(2,5)=1 або o(2,5)=0

– відмітити блоки з 2 по 5 як зайняті/вільні.

16. Асоціативний масив з сортуванням. Забезпечити доступ к елементам за індексом та символьним представленням. Використання псевдозмінних для асоціативного масиву аналогічно використанню псевдозмінних для стека (див. варіант 1).
17. Текст (слова, абзаци, форматування). Об'єктна модель тексту містить колекції Абзаців, Слів, Символів та відповідні методи роботи з ними: копіювання, вирізка/вставка, вирівнювання тексту в межах абзацу.
    Приклад псевдозмінних:
    String s="Hello";
    s(3,2)="p";
    В результаті рядок s буде дорівнювати "Help"(починаючи з третього символу, два сим воли заміняються на "p");
18. Віконний інтерфейс (відношення власник/підлеглий).
    Створити клас «Window» «Resizable Window» - з функціями відкрити, закрити, розвернути вікно, поміняти положення вікна. Можливість створення інтерфейсу – в якому для головного вікна можуть бути підлеглі вікна (приклад Access та ін..). Візуалізація – в текстовому режимі відображення всіх вікон.
19. Інтерфейс типу ієрархічного повноекранного меню. Створити класи роботи з меню, з можливістю ієрархії пунктів меню. Діалогове керування забезпечити за допомогою створеного класу повноекранного меню. Використання псевдозмінних: m1(2,5)=m2, де m1, m2 – пункти меню, тоді замінити з 2 по 5 пункти меню m1 пунктом меню m2.
20. Реляційні бази даних (зв'язки між таблицями). Реляційна база даних це сукупність «таблиць». Рядки утворюють «записи», стовпчики утворюють «поля записів», що можуть зберігати значення різних типів. Поняття «зв'язок» між таблицями застосовується в запитах до бази даних для визначення маршруту інформаційного пошуку. Для кожної пари об'єктів - таблиць може бути декілька об'єктів зв'язків.
    Таблиці: T1, T2,…,Tn
    Зв'язки: L1,L2,…,Ln
    Запит користувача:T1 L1 T2 – результат: всі записи з T1 та всі записи з T2, які відповідають записам з T1.
    Варіанти запитів користувача:
    T1 L2 T3
    T1 L1 T2 L2 T3 L4 T4
    Надати користувачеві можливість створення таблиць за структурою, зв'язків. Заповнення таблиць. Створити клас „База даних" з деякими фіксованими типами полів бази даних. Організувати зв'язки між таблицями по деякому фіксованому полю. Використання псевдозмінних: bd(3,5)=s, де bd – база даних, s – елемент бази даних, тоді з 3 по 5 елементи бази даних замінити на елемент s.
21. Реляційні бази даних (індексування). Реляційна база даних це сукупність «таблиць». Рядки утворюють «записи», стовпчики утворюють «поля записів», що можуть зберігати значення різних типів. Переупорядкування даних занадто громіздка операція. Замість переупорядкування записів (замість фізичного переупорядкування) застосовують «індекси». Об'єкт «індекс» для даної таблиці ставить у відповідність полю таблиці послідовність номерів записів. Крізь призму того чи іншого «індексу» таблиця виглядає відповідним чином упорядкованою при створенні індексу. Послідовність номерів поновлюється автоматично. Створити клас „База даних" з деякими фіксованими типами полів бази даних, організувати роботу з базами даних

(додавання записів, пошук запису, перехід до наступного запису, перехід до попереднього запису, перехід в кінець, перехід на початок). Використання псевдозмінних: bd(3,5)=s, де bd – база даних, s – елемент бази даних, тоді з 3 по 5 елементи бази даних замінити на елемент s.

22. Мережеві бази даних (введення/виведення, навігація). База даних представлена у вигляді орграфу.

Мережева модель даних визначається в тих же термінах, що й ієрархічна (п.23). Вона складається з множини записів, що можуть бути власниками чи членами групових відносин. Зв'язок між записом-власником і записом-членом також має вид 1:N.

Основне розходження цих моделей полягає в тому, що в мережевій моделі запис може бути членом більш ніж одного групового відношення. Відповідно до цієї моделі кожне групове відношення іменується і проводиться розходження між його типом і екземпляром. Тип групового відношення задається його ім'ям і визначає властивості загальні для всіх екземплярів даного типу. Екземпляр групового відношення представляється записом-власником і безліччю (можливо порожнім) підлеглих записів. При цьому мається наступне обмеження: екземпляр запису не може бути членом двох екземплярів групових відносин одного типу .

Операції над даними.

ДОДАТИ - внести запис у БД і, у залежності від режиму включення, або включити її в групове відношення, де вона оголошена підлеглої, або не включати ні в яке групове відношення.

ВКЛЮЧИТИ В ГРУПОВЕ ВІДНОШЕННЯ - зв'язати існуючу підлеглий запис із записом-власником.

ПЕРЕКЛЮЧИТИ - зв'язати існуючу підлеглий запис з інший записом-власником у тім же груповому відношенні.

ОБНОВИТИ - змінити значення елементів попередньо витягнутого запису.

ВИТЯГТИ - витягти запису послідовно за значенням ключа, а також використовуючи групові відносини - від власника можна перейти до записів - членам, а від підлеглого запису до власника набору.

ВИДАЛИТИ - забрати з БД запис. Якщо цей запис є власником групового відношення, то аналізується клас членства підлеглих записів. Обов'язкові члени повинні бути попередньо виключені з групового відношення, фіксовані вилучені разом із власником, необов'язкові залишаться в БД.

ВИКЛЮЧИТИ З ГРУПОВОГО ВІДНОШЕННЯ - розірвати зв'язок між записом-власником і записом-членом.

23. Ієрархічні бази даних (введення/виведення, навігація). База даних представлена у вигляді дерева. Організація даних у СУБД ієрархічного типу визначається в термінах: **елемент, агрегат, запис (група), групове відношення, база даних**. **Атрибут** (елемент даних) - найменша одиниця структури даних. Звичайно кожному елементу при описі бази даних привласнюється унікальне ім'я. По цьому імені до нього звертаються при обробці. Елемент даних також часто називають полем. **Запис** - іменована сукупність атрибутів. Використання записів дозволяє за одне звертання до бази одержати деяку логічно зв'язану сукупність даних. Саме записи змінюються, додаються і віддаляються. Тип запису визначається складом її атрибутів. Екземпляр запису - конкретний запис з конкретним значенням елементів. **Групове відношен**ня - ієрархічне відношення між записами двох типів. Батьківський запис (власник групового відношення) називається

вихідним записом, а дочірні записи (члени групового відношення) - підлеглими. Ієрархічна база даних може зберігати тільки такі деревоподібні структури.
Кореневий запис кожного дерева обов'язково повинне містити ключ з унікальним значенням. Ключі некореневих записів повинні мати унікальне значення тільки в рамках групового відношення. Кожен запис ідентифікується повним зчепленим ключем, під яким розуміється сукупність ключів усіх записів від кореневої по ієрархічному шляху.

Для групових відносин в ієрархічній моделі забезпечується автоматичний режим включення і фіксоване членство. Це означає, що для запам'ятовування будь-якого некореневого запису в БД повинна існувати її батьківський запис. При видаленні батьківського запису автоматично віддаляються всі підлеглі.
Операції над даними, визначені в ієрархічній моделі:
ДОДАТИ в базу даних новий запис. Для кореневого запису обов'язкове формування значення ключа.
ЗМІНИТИ значення даних попередньо витягнутого запису. Ключові дані не повинні піддаватися змінам.
ВИДАЛИТИ деякий запис і всі підлеглі їй запису.
ВИТЯГТИ: витягти кореневий запис за ключовим значенням, допускається також послідовний перегляд кореневих записів
витягти наступний запис (наступний запис витягається в порядку лівостороннього обходу дерева)

24. Орграфи: представлення та перетворення представлень.
Способи представлення графа в інформатиці:
Матриця суміжності

Матриця суміжності - таблиця, де як у стовпці, так і рядка відповідають вершинам графа. У кожного осередку цієї матриці записується число, що визначає наявність зв'язку від вершини-рядка до вершини-стовпця (або навпаки).

Недоліком є вимоги до пам'яті - очевидно, квадрат кількості вершин.

Матриця інцидентності

Кожен рядок відповідає визначеній вершині графа, а стовпці відповідають зв'язкам графа. В осередок на перетинанні i-ої рядка з j-м стовпцем матриці записується 1 у випадку якщо зв'язок j "виходить" з вершини i, -1 якщо зв'язок "входить" у вершину.

Даний спосіб є самим ємної (розмір пропорційний | E | | V |) і незручним для збереження, але полегшує перебування циклів у графі

Список ребер

Список ребер - це тип представлення графа в пам'яті, що припускає, що кожне ребро представляється двома числами - номерами вершин цього ребра. Список ребер більш

зручний для реалізації різних алгоритмів на графах у порівнянні з матрицею суміжності.

25. Текстові файли. Створити клас для роботи з текстовими файлами : Методи створення, видалення, переносу, редагування, перегляду. Властивості: атрибути (тільки для читання, архівний, скритий), дата створення, власник.

26. Плоскі геометричні фігури (візуалізація, перетворення координат (поворот, осева симетрія, паралельний перенос)).

27. Тривимірні геометричні фігури (візуалізація, перетворення координат (поворот, осева симетрія, паралельний перенос)).

28. Електроні таблиці. Створити класи роботи з електронними таблицями клас «Книга», клас «Лист», «Комірка». Методи: відкрити, закрити, перейменувати, занести значення в комірку, отримати значення комірки. Властивості: Атрибути, Захист (аналог – Excel – захистити робочу книгу або лист), Найменування.
29. Браузер. Модель www. Повноекранний текстовий режим.
30. Файл-менеджер. Операції над файлами і папками. Повноекранний текстовий режим.
31. Калькулятор. Основні операції над числами і пам’яттю. Повноекранний текстовий режим.