Міністерство освіти і науки України

Київський національний університет

імені Тараса Шевченка

**Методичні вказівки**

**до виконання лабораторних робіт**

**з курсу «Алгоритми і структури даних»**

для студентів, які навчаються за спеціальністю

122 «Комп’ютерні науки»

освітньої програми «Прикладне програмування»

Затверджено

редакційно-видавничою радою

університету, протокол №

від \_\_.\_\_\_\_\_\_\_\_\_\_.2019р.

Київ

КНУ ім. Т.Шевченка

2019

Методичні вказівки до виконання лабораторних робіт з курсу «Алгоритми і структури даних» для студентів, які навчаються за спеціальністю 122 «Комп’ютерні науки», освітньої програми «Прикладне програмування» / укл. В.Л. Плескач, М.В. Пирог. - Київ : КНУ ім. Т.Шевченка, 2019. - 44 с.

|  |  |
| --- | --- |
| Укладачі: | В.Л. Плескач |
|  | М.В. Пирог |
|  |  |
| Рецензент | В.В. Гавриленко |

Кафедра прикладних інформаційних систем

**ЗМІСТ**

[Вступ 4](#_Toc1)

[Загальні рекомендації щодо виконання лабораторних робіт 5](#_Toc2)

[Лабораторна робота №1. Лінійні алгоритми та алгоритми розгалуження 6](#_Toc3)

[1.1. Теоретичний матеріал 6](#_Toc4)

[1.2. Завдання для виконання 7](#_Toc5)

[Лабораторна робота №2. Одновимірні та двовимірні масиви 10](#_Toc6)

[2.1. Теоретичний матеріал 10](#_Toc7)

[2.2. Завдання для виконання 11](#_Toc8)

[Лабораторна робота №3. Структури даних та масиви структур 14](#_Toc9)

[3.1. Теоретичний матеріал 14](#_Toc10)

[3.2. Завдання для виконання 15](#_Toc11)

[Лабораторна робота №4. Однозв’язні та двозв’язні списки 18](#_Toc12)

[4.1. Теоретичний матеріал 18](#_Toc13)

[4.2. Завдання для виконання 19](#_Toc14)

[Лабораторна робота №5. Циклічні списки та покажчики 21](#_Toc15)

[5.1. Теоретичний матеріал 21](#_Toc16)

[5.2. Завдання для виконання 21](#_Toc17)

[Лабораторна робота №6. Структури стеки та черги 23](#_Toc18)

[6.1. Теоретичний матеріал 23](#_Toc19)

[6.2. Завдання для виконання 24](#_Toc20)

[Лабораторна робота №7. Дерева та сортування за допомогою дерев 26](#_Toc21)

[7.1. Теоретичний матеріал 26](#_Toc22)

[7.2. Завдання для виконання 26](#_Toc23)

[Лабораторна робота №8. Пошук та сортування одновимірних масивів 30](#_Toc24)

[8.1. Теоретичний матеріал 30](#_Toc25)

[8.2. Завдання для виконання 30](#_Toc26)

[Лабораторна робота №9. Структури графів, алгоритми Дейкстри та Флойда 32](#_Toc27)

[9.1. Теоретичний матеріал 32](#_Toc28)

[9.2. Завдання для виконання 33](#_Toc29)

[Лабораторна робота №10. Орієнтовані графи 36](#_Toc30)

[10.1. Теоретичний матеріал 36](#_Toc31)

[10.2. Завдання для виконання 36](#_Toc32)

[Лабораторна робота №11. Рекурсивні алгоритми 39](#_Toc33)

[11.1. Теоретичний матеріал 39](#_Toc34)

[11.2. Завдання для виконання 39](#_Toc35)

[Лабораторна робота №12. Незв’язні динамічні структури 41](#_Toc36)

[12.1. Теоретичний матеріал 41](#_Toc37)

[12.2. Завдання для виконання 41](#_Toc38)

[Рекомендовані джерела 43](#_Toc39)

# Вступ

В контексті підготовки фахівців з програмування на першому курсі закладів вищої освіти методичні вказівки до лабораторних робіт з дисципліни “Алгоритми і структури даних” є частиною загального методичного забезпечення курсу дисципліни “Алгоритми і структури даних”.

Метою виконання лабораторних робіт з дисципліни є закріплення та поглиблення набутих теоретичних знань з ціллю їх успішного застосування при розв’язанні практичних задач при створені програмного забезпечення.

Методичні вказівки місять завдання для 12 лабораторних робіт, необхідний для їх виконання теоретичний матеріал та варіанти завдань.

# Загальні рекомендації щодо виконання лабораторних робіт

На початку навчального семестру кожний студент академічної групи отримує від викладача завдання на виконання лабораторних робіт. За результатом виконання кожної лабораторної роботи студент формує звіт. Здача лабораторних робіт відбувається протягом усього навчального семестру відповідно до графіку їх проведення. Строк здачі комплексу лабораторних робіт - до підсумкового завдання. План проведення лабораторних робіт представлено нижче.

|  |  |
| --- | --- |
| Назва лабораторної роботи | Кількість годин |
| Лабораторна робота №1. Лінійні алгоритми та алгоритми розгалуження | 2 |
| Лабораторна робота №2. Одновимірні та двовимірні масиви | 2 |
| Лабораторна робота №3. Структури даних та масиви структур | 2 |
| Лабораторна робота №4. Однозв’язні та двозв’язні списки | 2 |
| Лабораторна робота №5. Циклічні списки та покажчики | 2 |
| Лабораторна робота №6. Структури стеки та черги | 2 |
| Лабораторна робота №7. Дерева та сортування за допомогою дерев | 2 |
| Лабораторна робота №8. Пошук та сортування одновимірних масивів | 2 |
| Лабораторна робота №9. Структури графів, алгоритми Дейкстри та Флойда | 2 |
| Лабораторна робота №10. Орієнтовані графи | 4 |
| Лабораторна робота №11. Рекурсивні алгоритми | 4 |
| Лабораторна робота №12. Незв’язні динамічні структури | 4 |

Завдання можна виконувати мовою програмування на вибір:

* C++;
* C#;
* Java.

Програму можна демонструвати у будь-якій операційній системі. Рекомендовано розробляти застосунок так, щоб його можна було зібрати та виконувати на широкому класі систем (цього елементарно можна досягнути переліченими мовами).

Вибір середовища розробки може бути продиктований вподобаннями студента. Доцільно використати декілька середовищ для різних робіт.

# Лабораторна робота №1. Лінійні алгоритми та алгоритми розгалуження

*Мета роботи:* дослідити та оволодіти навичками написання програм, що використовують лінійний алгоритми та алгоритми розгалуження.

## 1.1. Теоретичний матеріал

Під **алгоритмом** розуміється точне розпорядження, що задає послідовність дій, яка веде від довільного вихідного даного (або від деякої сукупності можливих для даного алгоритму вихідних даних) до досягнення повністю визначається цим вихідними даними результату.

Алгоритм повинен мати певні властивості, наявність яких гарантує отримання рішення задачі виконавцем.

**Дискретність.** Рішення завдання має бути розбите на елементарні дії. Запис окремих дій реалізується у вигляді впорядкованої послідовності окремих команд, що утворюють дискретну структуру алгоритму. Це властивість безпосередньо відображено у визначенні алгоритму.

**Зрозумілість.** На практиці будь-який алгоритм призначений для певного виконавця, і будь-яку команду алгоритму виконавець повинен вміти виконати.

**Детермінованість.** Кожна команда алгоритму повинна визначати однозначні дії виконавця. Результат їх виконання не повинен залежати від факторів, не врахованих в алгоритмі явно. при одних і тих самих вихідних даних алгоритм повинен давати стабільний результат.

**Масовість.** Розроблений алгоритм повинен давати можливість отримання результату при різних вихідних даних для однотипних завдань.

Базова структура **«проходження»** означає, що кілька операторів виконуються послідовно один за одним, і тільки один раз за час виконання програми. Структура «проходження» використовується для реалізації завдань, що мають лінійний алгоритм рішення.

Це означає, що такий алгоритм не містить перевірок умов і повторень, дії в ньому виконуються послідовно, одна за одною.

Другою базовою структурою є **«розгалуження»**. Ця структура забезпечує, в залежності від результату перевірки умови, вибір одного з альтернативних шляхів роботи алгоритму, причому кожен із шляхів веде до загального виходу, так що робота алгоритму буде тривати незалежно від того, який шлях буде обраний.

Існує структура з повним і неповним розгалуженням.

Структура з повним розгалуженням (якщо - то - інакше) записується так:

Якщо <умова>

то <дія 1>

інакше <дії 2>

все якщо

Команда виконується так:

якщо <умова> є істинним, то виконуються <дія 1>, записана після ключового слова то, якщо <умова> є хибною, то виконуються <дія 2>, записана після слова інакше.

Структура з неповним розгалуженням (якщо - то) не містить частини, що починається зі слова інакше:

Якщо <умова>

то <дії 1>

все якщо

Команда виконується так:

якщо <умова> є істинним, то виконуються <дії 1>, записані після ключового слова то.

## 1.2. Завдання для виконання

Завдання обраного варіанту реалізувати зі створенням меню перемикання між задачами. Змінні, що необхідні для розв’язку - вводити з клавіатури.

|  |  |
| --- | --- |
| 1. | 1. Швидкість човна в стоячій воді V км/год., швидкість течії річки U км/год. (U < V). Час руху човна по озеру T1 год., а по річці (проти течії ) T2 год. Визначити шлях S, що пройде човен.  2. Обчислити значення у, де .  3. Дано координати двох різних полів шахової дошки. Якщо королева за один хід може перейти з одного поля на інше, вивести логічне значення True, інакше вивести значення False. |
| 2. | 1. Швидкість першого автомобіля V1 км/год., другого – V2 км/год., відстань між ними S км. Визначити відстань між ними через Т годин, якщо автомобілі віддаляються один від інших.  2. Обчислити значення у, де .  3. Дано координати двох різних полів шахової дошки. Якщо кінь за один хід може перейти з одного поля на інше, вивести логічне значення True, інакше вивести значення False. |
| 3. | 1. Швидкість першого автомобіля V1 км/год., другого – V2 км/год., відстань між ними S км. Визначити відстань між ними через Т годин, якщо автомобілі рухаються назустріч один одному.  2. Обчислити значення у, де .  3. Дано координати двох різних полів шахової дошки. Якщо ферзь за один хід може перейти з одного поля на інше, вивести логічне значення True, інакше вивести значення False. |
| 4. | 1. Швидкість човна в стоячій воді V км/год., швидкість течії річки U км/год. (U < V). Час руху човна по озеру T1 год., а по річці (за течією течії) T2 год. Визначити шлях S, що пройде човен.  2. Обчислити значення у, де .  3. Дано координати двох різних полів шахової дошки. Якщо слон за один хід може перейти з одного поля на інше, вивести логічне значення True, інакше вивести значення False. |
| 5. | 1. Знайти площу кільця, внутрішній радіус якого дорівнює R1, а зовнішній радіус дорівнює R2 (R1 < R2).  2. Обчислити значення у, де +2.  3. Перевірити істинність твердження: «Дане ціле число є парним двозначним числом». |
| 6. | 1. Знайти периметр та площу прямокутної трапеції з основами a та b (a > b) та гострим кутом alpha (кут дано в радіанах).  2. Обчислити значення у, де .  3. Перевірити істинність твердження: «Серед трьох даних цілих чисел є хоча б одна пара чисел, що збігаються». |
| 7. | 1. Знайти відстань між двома точками, що задано координатами (x1, y1) та (x2, y2).  2. Обчислити значення у, де .  3. Перевірити істинність твердження: «Сума двох перших цифр даного чотиризначного числа дорівнює сумі двох його останніх цифр». |
| 8. | 1. Дано координати трьох вершин трикутника (x1, y1), (x2, y2), (x3, y3). Знайти його периметр та площу.  2. Обчислити значення у, де .  3. Перевірити істинність твердження: «Всі цифри даного тризначного числа різні». |
| 9. | 1. Дано координати чотирьох вершин чотирикутника (x1, y1), (x2, y2), (x3, y3), (x4, y4). Знайти його периметр та площу.  2. Обчислити значення у, де .  3. Перевірити істинність твердження: «Цифри даного тризначного числа утворюють арифметичну прогресію». |
| 10. | 1. Знайти відстань між двома точками, що задано координатами (x1, y1) та (x2, y2).  2. Обчислити значення у, де . Значення змінної х введіть з клавіатури.  3. Перевірити істинність твердження: «Цифри даного тризначного числа утворюють геометричну прогресію». |

**Контрольні питання:**

1. Структурування алгоритмів.
2. Структурування даних.
3. Інкапсуляція, як засіб структуризації.
4. Концепція структур даних.
5. Класифікація структур даних.
6. Базові операції над структурами даних.

# Лабораторна робота №2. Одновимірні та двовимірні масиви

Мета: дослідити та оволодіти навичками написання програм, пов’язаних із масивами.

## 2.1. Теоретичний матеріал

**Масив** - це впорядкований скінченний набір даних одного типу, які зберігаються в послідовно розташованих комірках оперативної пам'яті і мають спільну назву.

Масив складається з елементів. Кожен елемент має індекси, за якими його можна знайти у масиві. Кількість індексів визначає розмірність масиву. Розрізняють одно- та багатовимірні масиви.

**Одновимірний масив** це масив, доступ до елементів якого визначається за допомогою одного індексу.

Загальний вигляд конструкції опису одномірного масиву такий:

<тип> <ім'я масиву>[<розмір>]

Розмір - це кількість елементів масиву. Розмір масиву необхідно знати і задавати заздалегідь, оскільки компілятор має зарезервувати для нього необхідний обсяг пам'яті. Розміром може бути лише стала величина (не змінна).

int b[100]

int a = 1000; int b[a]; //помилка

**Двовимірний масив** це список одновимірних масивів.

Для доступу до елементів двовимірного масиву потрібно вказати два індекси. Якщо розглядати масив як таблицю, тоді перший індекс визначає рядок. Другий індекс визначає стовпець таблиці.

Загальний вигляд конструкції опису двовимірного масиву такий:

<тип> <ім'я масиву>[<довжина>][<висота>]

Для звернення до елемента двовимірного масиву слід вказати ім’я масиву і значення індексів у квадратних дужках. Наприклад, для звернення до елемента, розташованого у третьому рядку четвертого стовпця слід вказати: mas[3,4].

Ім'я масиву у програмі змінювати не можна воно є сталою величиною, що містить адресу першого елемента. Отже, назва масиву є вказівником на перший елемент.

## 2.2. Завдання для виконання

Завдання обраного варіанту реалізувати зі створенням меню перемикання між задачами. Змінні, що необхідні для розв’язку - вводити з клавіатури. Для заповнення масиву використовувати генератор випадкових чисел, якщо інше не вказано у завданні.

|  |  |
| --- | --- |
| 1. | 1. В одновимірному масиві розмірності N обчислити:  – кількість додатних та від’ємних елементів;  – добуток елементів, що розташовані між максимальним і мінімальним елементами.  2. Знайти в масиві зростаючі серії елементів, що мають довжину не менше трьох, вивести на екран масив та результат.  3. У прямокутній цілочисельній матриці визначити:  – кількість рядків, що не містять жодного нульового елемента;  – максимальне число, що зустрічається в матриці понад один раз.  4. Квадратну матрицю розмірності N заповнити одиницями вище основної діагоналі та нулями - нижче. |
| 2. | 1. В одновимірному масиві розмірності N обчислити:  – кількість парних та непарних елементів;  – суму елементів, що розташовані між максимальним і мінімальним елементами.  2. Знайти в масиві найдовшу спадаючу серію елементів.  3. У прямокутній цілочисельній матриці визначити:  – кількість рядків, що не містять жодного від’ємного елемента;  – мінімальне число, що зустрічається в матриці понад один раз.  4. Квадратну матрицю розмірності N заповнити нулями вище основної діагоналі та одиницями - нижче. |
| 3. | 1. В одновимірному масиві розмірності N обчислити:  – суму елементів з парними індексами;  – кількість елементів, що розташовані між першим і останнім нульовими елементами.  2. Впорядкувати масив перемістивши на початок всі додатні елементи в порядку зростання, в кінець масиву всі від’ємні елементи в порядку зростання, нулі помістити між додатними та від’ємними елементами.  3. У прямокутній цілочисельній матриці визначити:  – кількість рядків, що містять хоча б один нульовий елемент;  – номер рядка з найдовшою серією однакових елементів.  4. Квадратну матрицю розмірності N заповнити одиницями у верхньому трикутнику між діагоналями матриці, решту заповнити нулями. |
| 4. | 1. В одновимірному масиві розмірності N обчислити:  – суму елементів з непарними індексами;  – кількість елементів, що розташовані між першим і останнім від’ємними елементами.  2. Впорядкувати масив перемістивши на початок всі додатні елементи в порядку спадання, в кінець масиву всі від’ємні елементи в порядку спадання, нулі помістити між додатними та від’ємними елементами.  3. У прямокутній цілочисельній матриці визначити:  – кількість рядків, що містять хоча б один від’ємний елемент;  – номер рядка з найкоротшою серією однакових елементів.  4. Квадратну матрицю розмірності N заповнити одиницями у нижньому трикутнику між діагоналями матриці, решту заповнити нулями. |
| 5. | 1. В одновимірному масиві розмірності N обчислити:  – добуток елементів з парними індексами;  – кількість елементів, що розташовані між першим і останнім найбільшими елементами за модулем.  2. Видалити з масиву елементи, що знаходяться між заданими індексами. Місця, що звільнились - заповнити нулями в кінці масиву.  3. У прямокутній цілочисельній матриці визначити:  – суму у рядках, що мають непарний індекс;  – номер рядка з найбільшим добутком.  4. Квадратну матрицю розмірності N заповнити одиницями у лівому трикутнику між діагоналями матриці, решту заповнити нулями. |
| 6. | 1. В одновимірному масиві розмірності N обчислити:  – добуток елементів з непарними індексами;  – кількість елементів, що розташовані між максимальним і мінімальним елементами за модулем.  2. Замінити елементи, що знаходяться у масиві між заданими індексами одиницями.  3. У прямокутній цілочисельній матриці визначити:  – суму у рядках, що мають парний індекс;  – номер рядка з найменшим добутком.  4. Квадратну матрицю розмірності N заповнити одиницями у правому трикутнику між діагоналями матриці, решту заповнити нулями. |
| 7. | 1. В одновимірному масиві розмірності N обчислити:  – модулі елементів з парними індексами;  – суму елементів, що розташовані між заданими елементами.  2. Перетворити матрицю таким чином щоб в її першій половині стояли елементи, що мають парні індекси, а в другій - непарні.  3. Прямокутну цілочисельну матрицю повернути на 90 за годинниковою стрілкою.  4. Квадратну матрицю розмірності N заповнити одиницями у правому та лівому трикутниках між діагоналями матриці, решту заповнити нулями. |
| 8. | 1. В одновимірному масиві розмірності N обчислити:  – модулі елементів з непарними індексами;  – добуток елементів, що розташовані між заданими елементами.  2. Перетворити матрицю таким чином щоб в її першій половині стояли елементи, що мають непарні індекси, а в другій - парні.  3. Прямокутну цілочисельну матрицю повернути на 90 проти годинникової стрілки.  4. Квадратну матрицю розмірності N заповнити одиницями у верхньому та нижньому трикутниках між діагоналями матриці, решту заповнити нулями. |
| 9. | 1. В одновимірному масиві розмірності N обчислити:  – суму модулів елементів з парними індексами;  – добуток елементів за модулем, що розташовані між заданими елементами.  2. Перетворити матрицю замінивши елементи з парними індексами нулями.  3. Прямокутну цілочисельну матрицю заповнити “равликом” від 1 до n за годинниковою стрілкою.  4. Квадратну матрицю розмірності N заповнити одиницями вище побічної діагоналі та нулями - нижче. |
| 10. | 1. В одновимірному масиві розмірності N обчислити:  – суму модулів елементів з непарними індексами;  – суму елементів за модулем, що розташовані між заданими елементами.  2. Перетворити матрицю замінивши елементи з непарними індексами нулями.  3. Прямокутну цілочисельну матрицю заповнити “равликом” від 1 до n проти годинникової стрілки.  4. Квадратну матрицю розмірності N заповнити нулями вище побічної діагоналі та одиницями - нижче. |

**Контрольні запитання**

1. Дані арифметичного типу.Що таке масив?
2. Які існують засоби ініціалізації одновимірного масиву?
3. Організація введення/виведення масиву.
4. Наведіть приклад введення/виведення одновимірного масиву.
5. Як розміщується одновимірний масив в пам’яті?
6. Що таке багатовимірний масив?
7. Як знайти мінімальний та максимальний елементи масиву?
8. Як вказівка пов’язана з одновимірним масивом? Використання арифметичних операцій з вказівками.
9. Яким чином перетворити двовимірний масив до одновимірного?
10. Яким чином перетворити одновимірний масив до двовимірного?

# Лабораторна робота №3. Структури даних та масиви структур

Мета: оволодіти практичними навичками використання структур і масивів структур, навчитися складати програми для виконання операцій з полями структур, навчитися використовувати динамічне виділення пам’яті.

## 3.1. Теоретичний матеріал

**Структура даних** - загальна властивість інформаційного об'єкта, з яким взаємодіє та або інша програма. Ця загальна властивість характеризується:

* множиною допустимих значень цієї структури;
* набором допустимих операцій;
* характером організованості.

Найпростіші структури даних називаються також типами даних.

У програмуванні та комп'ютерних науках структури даних - це способи організації даних у комп'ютерах. Часто разом зі структурою даних пов'язується і специфічний перелік операцій, які можуть бути виконаними над даними, організованими в таку структуру.

Правильний підбір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів їх опрацювання. Добре побудовані структури даних дозволяють оптимізувати використання машинного часу та пам'яті комп'ютера для виконання найбільш критичних операцій.

Відома формула **“Програма = Алгоритми + Структури даних”** дуже точно виражає необхідність відповідального ставлення до такого підбору. Тому іноді навіть не обраний алгоритм для опрацювання масиву даних визначає вибір тієї чи іншої структури даних для їх збереження, а навпаки.

Розрізняють наступні рівні описуваннявання даних:

* абстрактний (математичний) рівень;
* логічний рівень;
* фізичний рівень.

**Логічний рівень** – подання структури даного на тій чи іншій мові програмування.

**Фізичний рівень** - відображення у пам'ять комп'ютера інформаційного об'єкту відповідно до логічного описування.

Оскільки пам'ять комп'ютера обмежена, то виникають питання розподілу пам'яті й керування нею.

Будь-яка структура на абстрактному рівні може бути подана у вигляді двійки <D,R>, де D - скінчена множина елементів, які можуть бути типами даних або структурами даних, a R - множина відношень, властивості якої визначають різні типи структур даних на абстрактному рівні.

Складеним типом даних назвемо тип даних, що складається із скінченної та наперед заданої множини елементів певного типу, які не обов'язково є атомарними.

Перерахуємо складені типи даних та дамо їм коротку характеристику.

**Множина** – скінчена сукупність елементів, в якої R = Ø.

**Послідовність** - абстрактна структура, у якої множина R складається з одного відношення лінійного порядку (тобто для кожного елемента, крім першого і останнього, є попередній і наступний елементи).

**Матриця** – структура, в якої множина R складається із двох відношень лінійного порядку.

**Дерево** – множина R складається з одного відношення ієрархічного порядку.

**Граф** – множина R складається з одного відношення бінарного порядку.

**Гіперграф** – множина R складається із двох і більше відношень різного порядку.

## 3.2. Завдання для виконання

При виконанні завдання також створити інтерфейс користувача та меню для роботи з програмою.

|  |  |
| --- | --- |
|  | Завдання |
| 1. | Описати структуру з ім’ям STUDENT, що містить наступні поля:  NAME – прізвище та ініціали, GROUP – номер групи, SES – оцінки з п’яти предметів (масив з п’яти елементів). Написати програму, що реалізовує наступні дії окремими функціями:  – введення з клавіатури даних у масив STUD, що складається з N змінних типу STUDENT;  – впорядкування записів за зростанням значень поля GROUP;  – виведення на екран прізвищ і номерів груп для всіх студентів, середній бал яких більший за 4.0; якщо таких студентів немає, то вивести відповідне повідомлення. |
| 2. | Описати структуру з ім’ям ABITURIENT, що містить наступні поля: NAME – прізвище, ініціали, GENDER – стать, SPEC – назва спеціальності, EXAM – результати вступних іспитів з трьох предметів (масив з трьох елементів). Написати програму, що окремими функціями реалізовує наступні дії:  – введення з клавіатури даних у масив ABITUR, що складається з N змінних типу ABITURIENT;  – впорядкування записів за зростанням середнього бала;  – виведення на екран прізвищ та назв спеціальностей для всіх абітурієнтів, що мають бал нижче, ніж прохідний, що визначається користувачем програми; якщо таких студентів немає, то вивести відповідне повідомлення. |
| 3. | Описати структуру з ім’ям SCHOOL, що містить наступні поля:  NAME – прізвище та ім’я учня, GROUP – номер класу, SUBJECT – успішність з п’яти предметів (масив з п’яти елементів). Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив LEARNER, що складається з N змінних типу SCHOOL;  – впорядкування записів за алфавітом;  – виведення на екран прізвищ і номерів груп для всіх студентів, що мають хоча б одну оцінку 2; якщо таких учнів немає, то вивести відповідне повідомлення. |
| 4. | Описати структуру з ім’ям AEROFLOT, що містить наступні поля: CITY – назва населеного пункту призначення, NUM – номер рейса, TYPE – тип літака. Написати програму, що окремими функціями реалізовує наступні дії:  – введення з клавіатури даних у масив AIR, що складається з N змінних типу AEROFLOT;  – впорядкування записів за зростанням номеру рейсу;  – виведення на екран номерів рейсів і типів літаків, що вилетіли в пункт призначення, назва якого збіглася з назвою, введеною з клавіатури; якщо таких рейсів немає, то вивести відповідне повідомлення. |
| 5. | Описати структуру з ім’ям SKLAD, що містить наступні поля: NAME – назва товару, TYPE – одиниця виміру товару, QUANTITY – кількість одиниць товару, COST – ціна одиниці товару. Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив SHOP, що складається з N змінних типу SKLAD;  – впорядкування записів за назвами товару;  – виведення на екран інформації про товар, його кількість, ціну одиниці та обчислену загальну суму на складі, назва якого вводиться з клавіатури; якщо такого немає, то вивести відповідне повідомлення. |
| 6. | Описати структуру з ім’ям WORKER, що містить наступні поля: NAME – прізвище та ініціали працівника, POS – назва посади, YEAR – рік прийняття на роботу, MONTH – місяць прийняття на роботу. Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив TABL, що складається з N змінних типу WORKER;  – впорядкування записів в алфавітному порядку;  – виведення на екран прізвищ працівників, стаж роботи яких перевищує значення, введене з клавіатури; якщо таких працівників немає, то вивести відповідне повідомлення. |
| 7. | Описати структуру з ім’ям TRAIN, що містить наступні поля: NAZV – назва пункту призначення, NUMR – номер потягу, DATE – дата відправлення, TIME – час відправлення. Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив RASP, що складається з N змінних типу TRAIN;  – впорядкування записів за алфавітом за назвами пунктів призначення;  – виведення на екран інформації про поїзди, що відправляються після введеного з клавіатури дня та часу. якщо таких поїздів немає, то вивести відповідне повідомлення. |
| 8. | Описати структуру з ім’ям TIMETABLE, що містить наступні поля: NAZV – назва пункту призначення, NUMR – номер поїзда, DATE – дата відправлення, TIME – час відправлення. Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив TRAIN, що складається з N змінних типу TIMETABLE;  – впорядкування записів за датою та часом відправлення поїзда;  – виведення на екран інформації про поїзди, що направляються в пункт призначення, назва якого введена з клавіатури, якщо таких поїздів немає, то вивести відповідне повідомлення. |
| 9. | Описати структуру з ім’ям ITINERARY, що містить наступні поля: FIRST – назва початкового пункту маршруту, FINAL – назва кінцевого пункту маршруту, NUM – номер маршруту, DISTANCE – відстань у кілометрах. Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив ROUT, що складається з N змінних типу ITINERARY;  – впорядкування записів за спаданням відстані у кілометрах;  – виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо таких маршрутів немає, то вивести відповідне повідомлення. |
| 10. | Описати структуру з ім’ям NOTE, що містить наступні поля: NAME – прізвище, ім’я, TEL – номер телефону, BDAY – день народження (масив із трьох чисел). Написати програму, що окремими функціями виконує наступні дії:  – введення з клавіатури даних у масив BLOCKNOTE, що складається з N змінних типу NOTE;  – впорядкування записів за зростанням дат днів народження;  – виведення на екран інформації про людей, чиї дні народження припадають на місяць, значення якого введено з клавіатури; якщо таких людей немає, то вивести відповідне повідомлення. |

**Контрольні запитання**

1. Поняття структури даних.
2. Рівні подання структур даних.
3. Рівні описування даних.
4. Основні види (типи) структур даних.
5. Класифікація структур даних у програмах користувача й у пам'яті комп'ютера.
6. Приклади структур даних.
7. Порівняти різні схеми зберігання даних.
8. Навести приклади прямого доступу до даних.

# Лабораторна робота №4. Однозв’язні та двозв’язні списки

Мета: навчитися складати програми для роботи зі списками.

## 4.1. Теоретичний матеріал

У комп'ютерній науці, список або послідовність являє собою абстрактний тип даних, який представляє собою зліченне число впорядкованих значень, де одне і теж саме значення може зустрічатися більше одного разу. Екземпляр списку - це комп'ютерне уявлення математичного поняття скінченної послідовності; (потенційно) нескінченний аналог списку - це потік. Списки є основним прикладом контейнерів, так як вони містять інші значення. Якщо ж значення зустрічається кілька разів, в кожному випадку воно буде розглядатися, як окремий елемент.

Назва «список» також використовується для кількох конкретних структур даних, які можуть бути використані для реалізації абстрактних списків, особливо зв'язаних списків.

Багато мов програмування забезпечують підтримку типів даних списку, і мають спеціальний синтаксис і семантику для списків і список операцій. Список часто можна побудувати шляхом написання елементів в послідовності, розділених комами, крапками з комою та/або пробілами, в межах пари роздільників, таких як круглі дужки '()', квадратні дужки '[]', фігурні дужки '{}', або кутові дужки '<>'. В об'єктно-орієнтованих мовах програмування, списки зазвичай надаються як екземпляри підкласів загального «списку» класу, і проходиться через окремі ітератори. Список типів даних часто реалізуються з використанням структури даних масиву або зв'язаних списків деякого виду, але і інші структури даних можуть бути більш відповідною для деяких застосувань. У деяких контекстах, наприклад, в програмуванні на LISP, термін список може ставитися конкретно до зв'язаного списку, а не масиву.

У теорії типів і функціональному програмуванні, абстрактні списки зазвичай індуктивно визначаються двома операціями: NIL, що дає порожній список, і CONS, яка додає елемент у початок списку.

Однозв’язний список - вид зв'язаного списку, який складається з вузлів, кожен з яких містить у собі данні (інформаційну частину) та посилання на наступний вузол.

Найчастіше вузлом списку вважають структурний тип (структуру), який зберігає у собі певну інформаційну частину (іншу структуру або тип даних) та посилання (вказівник) на наступний вузол у списку. Список має «голову» head, тобто вказівник на початок списку та інколи має кінець tail, проте найчастіше його не використовують.

Двозв’язний список - вид зв'язаного списку, у якому посилання в кожному вузлі вказують на попередній і на подальший вузол у списку.

## 4.2. Завдання для виконання

***Завдання №1:***

В лабораторній роботі необхідно організувати однонаправлений список із структурою взятої із попередньої лабораторної роботи згідно з варіантом та передбачити такі дії :

• додавання елементів до списку;

• вилучення елементів зі списку;

• додавання елементів у кінець списку;

• пошук за заданим полем;

• друкування списку.

***Завдання №2.***

|  |  |
| --- | --- |
| № | Завдання |
| 1 | Дано кільцевий список, що містить 20 прізвищ гравців футбольної команди. Розбити гравців на 2 групи по 10 чоловік. У другу групу потрапляє кожен 12-й чоловік. |
| 2 | Дано 2 кільцевих списки, що містять прізвища спортсменів 2-х фехтувальних команд. Провести жеребкування. У першій команді вибирається кожен n-й гравець, а в другій - кожен m-й. |
| 3 | Задача Джозефуса. |
| 4 | Дано 2 кільцевих списки, що містять прізвища учасників лотереї та найменування призів. Виграє N чоловік (кожен К-й). Число для перерахунку призів – t. |
| 5 | Дано 2 списки, що містять прізвища учнів та номери екзаменаційних білетів. Число перерахунку для квитків – Е, для учнів – К. Визначити номери квитків, що будуть витягнені учнями |
| 6 | Дано список, що містить перелік товарів. З елементів 1-го списку (товари виготовлені фірмою SONY) створити новий список. |
| 7 | Дано 2 списки, що містять прізвища студентів 2-х груп. Перевести L студентів з 1-ї групи в другу. Число перерахунку-К. |
| 8 | Дано 2 списки, що містять перелік товарів, що виробляються концерном BOSH та PHILIPS. Створити список товарів, що випускаються як однієї так і іншою фірмою. |
| 9 | Дано 2 списки, що містять прізвища футболістів основного складу команди і запасного. Провести К замін. |
| 10 | Дано 2 списки, що містять прізвища солдатів 1-го і 2-го взводів. Під час атаки М солдатів з 1-го взводу загинули. Зробити поповнення солдатами 2-го взводу. |

**Контрольні запитання**

1. Дати визначення однозв’язному списку.
2. Дати визначення двозв’язкому списку.
3. Реалізація списку.
4. В чому різниця між статичним та динамічним списком?
5. Дати визначення кільцевого списку.
6. Дати визначення селекторів.

# Лабораторна робота №5. Циклічні списки та покажчики

Мета: опанування принципів розробки програм, що в основі використовують циклічні списки та взаємодію з покажчиками.

## 5.1. Теоретичний матеріал

Різновидом зв'язкових списків є кільцевої (циклічний, замкнутий) список. Він теж може бути однозв'язного або двозв’язного типу. Останній елемент кільцевого списку містить покажчик на перший, а перший (в разі двозв’язного списку) - на останній.

Як правило, така структура реалізується на базі лінійного списку. З кожним кільцевих списком додатково зберігається покажчик на перший елемент. У цьому списку посилання на NULL не зустрічається.

Також існують циклічні списки з виділеним головним елементом, що полегшують повний прохід через список.

## 5.2. Завдання для виконання

|  |  |
| --- | --- |
| № з\п | **Завдання** |
| 1 | Дано покажчик P1 на початок однозв'язного лінійного списку. Перетворити вихідний (однозв'язний) ланцюжок у двохзв’язний, в якому кожен елемент пов'язаний не тільки з подальшим елементом (за допомогою поля Next), але і з попереднім (за допомогою поля Prev). Поле Prev першого елемента покласти рівним nil. Вивести на екран перетворений ланцюжок у зворотному порядку. |
| 2 | Дано покажчик P1 на перший елемент не порожнього двохзв’язного списку. Продублювати в списку всі елементи з непарними значеннями (нові елементи додавати перед існуючими елементами з такими ж значеннями) і вивести покажчик на перший елемент перетвореного списку. |
| 3 | Дано покажчик P1 на перший елемент не порожнього двохзв’язного списку. Продублювати в списку всі елементи з непарними значеннями (нові елементи додавати після існуючих елементів з такими ж значеннями) і вивести покажчик на останній елемент перетвореного списку. |
| 4 | Дано покажчик P1 на перший елемент не порожнього двохзв’язного списку, що містить не менше двох елементів. Видалити зі списку всі елементи з непарними номерами і вивести покажчик на перший елемент перетвореного списку. Після видалення елементів із списку звільняти пам'ять, яку вони займали. |
| 5 | Дано покажчик P1 на перший елемент не порожнього двохзв’язного списку. Видалити зі списку всі елементи з непарними значеннями і вивести покажчик на перший елемент перетвореного списку (якщо в результаті видалення елементів список виявиться порожнім, то вивести nil). Після видалення елементів із списку звільняти пам'ять, яку вони займали. |
| 6 | Дано число K (> 0) і покажчик P0 на один з елементів не порожнього двохзв’язного списку. Перемістити в списку даний елемент на K позицій вперед (якщо після даного елемента знаходиться менше K елементів, то перемістити його в кінець списку). Вивести покажчики на перший і останній елементи перетвореного списку. Операції видалення і звільнення пам'яті не використовувати, поля з даними (Data) не змінювати. |
| 7 | Дано покажчик P1 на перший елемент не порожнього двохзв’язного списку. Перегрупувати його елементи, перемістивши всі елементи з непарними номерами в кінець списку (в тому ж порядку) і вивести покажчик на перший елемент перетвореного списку. Операції видалення і звільнення пам'яті не використовувати, поля з даними (Data) не змінювати. |
| 8 | Дано два не порожніх двохзв’язних списків і пов'язані з ними покажчики: PA і PB вказують на перший і останній елементи першого списку, PC - на один з елементів другого. Об'єднати вихідні списки, помістивши всі елементи першого списку (в тому ж порядку) після даного елемента другого списку, і вивести покажчики на перший і останній елементи об'єднаного списку. Операції видалення і звільнення пам'яті не використовувати. |
| 9 | Дано покажчики PX і PY на два різних елементи двохзв’язного списку; елемент з адресою PX знаходиться в списку перед елементом з адресою PY, але не обов'язково поруч з ним. Перемістити елементи, розташовані між даними елементами (не включаючи дані елементи) в новий список (в тому ж порядку). Вивести покажчики на перші елементи перетвореного і нового списків. Якщо новий список виявиться порожнім, то пов'язаний з ним покажчик покласти рівним nil. Операції видалення і звільнення пам'яті не використовувати. |
| 10 | Дано покажчики P1 і P2 на перший і останній елементи не порожнього двохзв’язного списку, що містить парну кількість елементів. Перетворити список в два циклічних списки, перший з яких містить першу половину елементів вихідного списку, а другий - другу половину. Вивести покажчики PA і PB на два середніх елемента вихідного списку (елемент з адресою PA повинен входити в перший циклічний список, а елемент з адресою PB - у другий). Операції видалення і звільнення пам'яті не використовувати. |

**Контрольні запитання**

1. Дати визначення покажчика в списку.
2. Дати визначення циклічного списку.
3. Методи роботи зі списками.

# Лабораторна робота №6. Структури стеки та черги

Мета: набути практичних навичок побудови алгоритмів, пов’язаних зі стеками, чергами.

## 6.1. Теоретичний матеріал

Стек – це колекція, елементи якої отримують за принципом “останній увійшов, перший вийшов” (Last – In – First – Out або LIFO). Це означає, що ми матимемо доступ тільки до останнього доданого елемента.

На відміну від списків, ми не можемо отримати доступ до довільного елемента стека. Ми можемо тільки додавати або видаляти елементи за допомогою спеціальних методів. У стека немає також методу Contains, як у списків. Крім того, у стека немає ітератора. Для того щоб розуміти, чому на стек накладаються такі обмеження, давайте подивимося на те, як він працює і як використовується.

Аналогія, що найчастіше зустрічається, для пояснення стека – стопка тарілок. Незалежно від того, скільки тарілок у стопці, ми завжди можемо зняти верхню. Чисті тарілки так само кладуться на верх стопки, і ми завжди будемо першими брати ту тарілку, яка була покладена останньою.

Якщо ми покладемо, наприклад, червону тарілку, потім синю, а потім зелену, то спочатку потрібно буде зняти зелену, потім синю, і, нарешті, червону. Головне, що потрібно запам’ятати – тарілки завжди ставляться і на верх стопки. Коли хтось бере тарілку, він також знімає її згори. Виходить, що тарілки розбираються в порядку, зворотному тому, в якому ставилися.

Тепер, коли ми розуміємо, як працює стек, введемо декілька термінів. Операція додавання елемента на стек називається “push”, видалення – “pop”. Останній доданий елемент називається верхівкою стека, або “top”, і його можна подивитися за допомогою операції “peek”. Давайте тепер подивимося на заготівлю класу, що реалізовує стек.

Черги дуже схожі на стеки. Вони також не дають доступу до довільного елемента, але, на відміну від стека, елементи кладуться (enqueue) і забираються (dequeue) з різних кінців. Такий метод називається “перший увійшов, перший вийшов” (First – In – First – Out або FIFO). Забирати елементи з черги ми будемо в тому ж порядку, що і ложили. Подібно до реальної черги або конвеєра.

Черги часто використовуються в програмах для реалізації буфера, в який можна покласти елемент для подальшої обробки, зберігаючи порядок вступу. Наприклад, якщо база даних підтримує тільки одно з’єднання, можна використати чергу потоків, які будуть, як не дивно, чекати своєї черги на доступ до БД.

## 6.2. Завдання для виконання

|  |  |
| --- | --- |
| № | **Завдання** |
| 1 | Створити стек цілочисельних значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (push) і видалення (pop) елемента зі стеку. Додайте в стек числа 4, 3, 1, 2, 4 і роздрукуйте вміст стека. Видаліть один елемент зі стеку, і роздрукуйте вміст стека ще раз. Знайдіть мінімальний елемент, що належить стеку. |
| 2 | Створити чергу дійсних значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (enqueue) та видалення (dequeue) елемента з черги. Додайте в чергу числа: -2.2,2.3, 2.2, 5.1, 6.7 і роздрукуйте вміст черги. Видаліть 3 елементи з черги, потім додайте в чергу число 1.9 і роздрукуйте чергу ще раз. Знайдіть похідні елементів, що належать черзі. |
| 3 | Створити стек строкових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (push) і видалення (pop) елемента з стека. Додайте в стек рядки «abc», «fx», «glc», «hi», «Gogo» і роздрукуйте вміст стека. Видаліть один елемент зі стеку, потім додайте рядок «the end» і роздрукуйте вміст стека ще раз. Знайдіть кількість рядків в стеці, що складаються з 2 символів. |
| 4 | Створити чергу строкових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (enqueue) і видалення (dequeue) елемента з черги. Додайте в чергу рядки «one», «two», «Three», «four» і роздрукуйте вміст черги. Видаліть 2 елементи з черги, потім додайте в чергу рядок «inf» і роздрукуйте чергу ще раз. Знайдіть сумарну довжину рядків, що належать черзі, крім першого рядка черзі. |
| 5 | Створити стек цілочисельних значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (push) і видалення (pop) елемента з стека. Додайте в стек числа 1, 2, 3, 4, 5 і роздрукуйте вміст стека. Видаліть 3 елементи з стека, і роздрукуйте вміст стека ще раз. Знайдіть суму елементів стека. |
| 6 | Створити чергу речових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (enqueue) і видалення (dequeue) елемента з черги. Додайте в чергу числа 2.1, 2.1, 5.3 і роздрукуйте вміст черги. Видаліть 1 елемент з черги, потім додайте в чергу число 4.9 і роздрукуйте чергу ще раз. Знайдіть суму елементів черги. |
| 7 | Створити чергу речових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (enqueue) і видалення (dequeue) елемента з черги. Додайте в чергу числа 2.2, 1.2, 2.0, 5.2 і роздрукуйте вміст черги. Видаліть 2 елементи з черги, потім додайте в чергу число 2.9 і роздрукуйте чергу ще раз. Знайдіть суму елементів черги. |
| 8 | Створити стек строкових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (push) і видалення (pop) елемента з стека. Додайте в стек рядки «sdf», «2», «ssd4», «hello» і роздрукуйте вміст стека. Видаліть 2 елементи з стека, і роздрукуйте вміст стека ще раз. Знайдіть рядок мінімальної довжини, що належить стеку. |
| 9 | Створити чергу строкових значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (enqueue) і видалення (dequeue) елемента з черги. Додайте в чергу рядки «one», «two», «three», «four» і роздрукуйте вміст черги. Видаліть 1 елемент з черги, потім додайте в чергу рядок «five» і роздрукуйте чергу ще раз. Знайдіть сумарну довжину всіх рядків, що належать черзі. |
| 10 | Створити стек цілочисельних значень, для реалізації використовуючи однозв'язні списки. Реалізувати операції додавання (push) і видалення (pop) елемента з стека. Додайте в стек числа -5, 3, -4, 5 і роздрукуйте вміст стека. Видаліть один елемент з стека, додайте число 10 в стек, і надрукуйте вміст стека ще раз. Знайдіть суму всіх позитивних елементів, що належать стеку. |

**Контрольні запитання**

1. Що таке черга?
2. Опишіть порядок роботи з чергою.
3. Які ви знаєте операції над чергами?
4. Що таке стек?
5. Опишіть порядок роботи зі стеком.

# Лабораторна робота №7. Дерева та сортування за допомогою дерев

Мета: дослідити та оволодіти навичками написання програм, що використовують при роботі із деревами та при сортуванні за допомогою дерев.

## 7.1. Теоретичний матеріал

Дерево - структура даних, яка складається з вузлів(вершин) і ребер, без будь-яких циклів. Дерево без вузлів називається нульовим або порожнім деревом. Дерево, яке не є порожнім, складається з кореневого вузла і багатьох рівнів додаткових вузлів, які утворюють ієрархію.

З визначення випливає, що кожна вершина є в свою чергу коренем деякого піддерева. Кількість піддерев вершини має назву ступеня (degree) цієї вершини. Вершина ступеню нуль має назву кінцевої (terminal) або листа (leaf). Некінцева вершина також має назву вершини розгалуження (branch node).

Нехай x - довільна вершина дерева з коренем r. Тоді існує єдиний шлях з r до x. Усі вершини на цьому шляху називаються предками (ancestors) x; якщо деяка вершина y є предком x, то x називається нащадком (descendant) y. Нащадки та предки вершини x, що не збігаються з нею самою, називаються власними нащадками та предками. Кожну вершину x, в свою чергу, можна розглядати як корінь деякого піддерева, елементами якого є вершини-нащадки x.

Якщо вершина x є предком y та не існує вершин поміж ними (тобто x та y з'єднані одним ребром), а також існують предки для x (тобто x не є коренем), то вершина x називається батьком (parent) до y, а y - дитиною (child) x. Коренева вершина єдина не має батьків.

Вершини, що мають спільного батька, називаються братами (siblings). Вершини, що мають дітей, називаються внутрішніми (internal). Глибиною вершини x називається довжина шляху від кореня до цієї вершини. Максимальна глибина вершин дерева називається висотою.

Якщо існує відносний порядок на піддеревах T1…Tm, то таке дерево називається впорядкованим (ordered tree) або пласким (plane tree).

Лісом називають множину дерев, які не перетинаються.

Найчастіше дерева в інформатиці зображують з коренем, який знаходиться зверху (говорять, що дерево в інформатиці «росте вниз»).

Важливим окремим випадком кореневих дерев є бінарні дерева, які широко застосовуються в програмуванні і визначаються як множина вершин, яка має виокремлений корінь та два піддерева (праве та ліве), що не перетинаються, або є пустою множиною вершин (на відміну від звичайного дерева, яке не може бути пустим).

## 7.2. Завдання для виконання

В першому завдання лабораторної роботи необхідно організувати дерево, передбачити виконання наступних дій (обхід дерева, пошук в бінарному дереві, включення елемента в дерево, видалення елемента із дерева) та виконати завдання згідно з варіантом. В другому завданні лабораторної роботи виконати сортування за допомогою дерева.

|  |  |
| --- | --- |
| № | Завдання |
|  | 1. Описати функцію, що:  a) присвоює параметру Е запис з самого лівого листа не порожнього дерева Т (лист-вершина, з якого не виходить ні однієї гілки);  б) визначає число входжень запису Е в дерево Т.  2. На заводі випустили деталі з наступними серійними номерами: 45, 56, 13, 75, 14, 18, 43, 11, 52, 12, 10, 36, 47, 9. Деталі з парними номерами надходять на склад № 1, а з непарними на склад № 2. Потрібно відсортувати деталі на складі № 1. |
|  | 1. Вершини дерева дійсні числа. Описати функцію, що:  a) обчислює середнє арифметичне всіх вершин дерева;  б) додає в дерево вершину зі значенням, обчисленим в попередній функції.  2. Угнали автомобіль. Свідок запам'ятав, що першою цифрою номера була 4. У базі викрадених автомобілів в цей день були наступні номери: 456, 124, 786, 435, 788, 444, 565, 127, 458, 322, 411, 531, 400, 546, 410. Потрібно скласти список номерів починаються на 4 і впорядкувати його за зростанням. |
|  | 1. Записи вершин дерева – дійсні числа. Описати функцію, що видаляє всі вершини з від’ємними записами.  2. За тиждень їзди в транспорті накопичилися квитки з номерами 124512, 342351, 765891, 453122, 431350, 876432, 734626, 238651, 455734, 234987. Потрібно відібрати "щасливі" квитки і розташувати їх за зростанням. |
|  | 1. Записи вершин дерева – дійсні числа. Описати функцію, що:  a) знаходить максимальне або мінімальне значення записів вершин не порожнього дерева;  б) друкує записи з усіх листів дерева.  2. Дано список людей із зазначенням їх віку. Для складання графіка відходу співробітників на пенсію потрібно скласти новий список в тому порядку, в якому вони будуть йти на пенсію |
|  | 1. Описати функцію, що:  a) визначає, чи входить вершина із записом Е в дерево Т;  б) якщо такий запис не знайдено, то додати його.  2. Студенти здали п'ять іспитів. Потрібно відсортувати список студентів за зростанням загального балу за результатами складених іспитів. |
|  | 1. Описати функцію, що:  a) знаходить в не порожньому дереві Т довжину (число гілок) шляху від кореня до найближчої вершини з записом Е; якщо Е не входить в Т, то за відповідь прийняти -1.  б) визначає максимальну глибину не порожнього дерева Т, тобто число гілок у самому довгому з шляхів від кореня дерева до листя.  2. У місті був один автобусний парк, куди приїжджали автобуси з номерами: 11, 32, 23, 12, 6, 52, 47, 63, 69, 50, 43, 28, 35, 33, 42, 56, 55, 101. Після будівництва другого автопарку вирішили перевести туди автобуси з непарними номерами. Для того щоб скласти розклад їх руху потрібно організувати список номерів автобусів другий парк, упорядкувавши їх за спаданням. |
|  | 1. Описати функцію ЕQUAL (T1, T2), що перевіряє на рівність дерева Т1 і Т2 (дерева рівні, якщо ключі і записи вершин одного дерева дорівнюють відповідно ключам і записам іншого дерева).  2. Була складена відомість по зарплаті, представлена у вигляді: Іванов – 166000, Сидоров – 180000, ... Потрібно упорядкувати цей список таким чином, щоб розмір зарплати зменшувався. |
|  | 1. Описати функцію, що:  a) друкує вузли не порожнього дерева при обході зліва направо;  б) видаляє всі листи вихідного дерева та друкує модифіковане дерево.  2. На стоянці стоять автомобілі з наступними номерами: 1212, 3451, 7694, 4512, 4352, 8732, 7326, 2350, 4536, 2387, 5746, 6776, 4316, 1324. Для статистики необхідно скласти список автомобілів з такими номерами, сума перших двох цифр яких дорівнює сумі двох останніх цифр, так щоб кожен наступний номер був менше попереднього. |
|  | 1. Описати функцію, що:  a) присвоює змінній b типу char значення: К – якщо вершина – корінь, П – якщо вершина – проміжна вершина, А – якщо вершина –лист;  б) роздруковує атрибути всіх вершин дерева.  2. Випустили лотерейні квитки з чотиризначними номерами. Виграшними вважаються ті квитки, сума цифр яких ділиться на 4. Скласти список виграшних квитків, впорядкованих за спаданням. |
|  | 1. Описати функцію, що:  а) вставляє вузол із записом Е в дерево, якщо раніше такого не було;  б) видалити його, якщо він вже існує.  2. Молода людина взяв номер телефону у своєї знайомої, але забув його. Він зміг пригадати тільки перші три цифри: 469 \*\*\*. У його записній книжці були наступні номери телефонів: 456 765, 469 465, 469 321, 616 312, 576 567, 469 563, 567 564, 469 129, 675 665, 469 873, 569 090, 469 999, 564 321, 469 010. Скласти список номерів починаються з цифр 469 і впорядкувати їх за спаданням. |

**Контрольні запитання**

1. Дати визначення бінарного дерева.
2. Операції над вузлами дерев.
3. Подібність дерев.
4. Властивості дерев.
5. Упорядкування дерев.
6. Методи обходу.

# Лабораторна робота №8. Пошук та сортування одновимірних масивів

Мета: набути практичних навичок застосування алгоритмів пошуку та сортування.

## 8.1. Теоретичний матеріал

Алгоритм пошуку — алгоритм, який вирішує задачу пошуку, тобто, знаходить інформацію, яка зберігається в певній структурі даних. Структури даних можуть бути реалізовані за допомогою зв'язаних списків, масивів, дерев пошуку, хеш-таблиць чи інших методів зберігання інформації. Алгоритм пошуку на пряму залежить від структури даних для якої він реалізований.

Алгоритми пошуку класифікуються на основі їх механізму пошуку. Лінійний пошук працює достатньо повільно, відносно бінарного пошуку, але на практиці часто зустрічається. Алгоритм лінійного пошуку порівнює кожний елемент в структурі даних зі значенням котре потрібно знайти. Існує також бінарний пошук, або половинно-інтервальний пошук, однак він працює лише з відсортованими елементами, один з найшвидших алгоритмів, який повторно порівнює пошуковий елемент з центральним елементом заданої структури, якщо знайшов то кінець, якщо ні, то порівнює елементи та продовжує пошук в лівій чи правій частині структури даних поки не знайде потрібний. Реалізація алгоритму пошуку для хеш-таблиці — складне завдання, бо в такій структурі даних кожному елементу відповідає певний ключ (key), та данні відсортовані по цим ключам. Потрібно досить гарно реалізувати хеш-функцію для пошуку елементу по ключу, бо алгоритм пошуку буде шукати спочатку ключ, а потім отримуватиме значення по ключу.

Алгоритми пошуку оцінюються на основі їх складності чи максимального часу виконання. Бінарний пошук має максимальну складність O ( log ( n ) ) {\displaystyle \mathrm {O} (\log(n))} {\displaystyle \mathrm {O} (\log(n))}, або логарифмічний час виконання. Це означає, що максимальна кількість операцій, необхідних для пошуку C m p max = log 2 ⁡ n {\displaystyle Cmp\_{\max }=\log \_{2}n} {\displaystyle Cmp\_{\max }=\log \_{2}n}, де n {\displaystyle n} n — кількість елементів.

## 8.2. Завдання для виконання

У задачі забезпечити пошук потрібних елементів у непосортованих масивах. Здійснити їх сортування.

Алгоритми пошуку:

* Лінійний пошук.
* Лінійний пошук з бар’єром.
* Бінарний пошук.
* Пошук Фібоначі.
* Пошук з перестановкою в початок.
* Пошук с транспозицією.

Алгоритми сортування:

* Сортування обміном
* Сортування вибором
* Бульбашкове сортування
* Сортування включениями
* Бульбашкове сортування вставками.
* Швидке сортування
* Сортування Шелла
* Пірамідальне сортування
* Сортування Хоара.

Знайти, а потім відсортувати:

|  |  |
| --- | --- |
| 1 | Елементи, які присутні в обох масивах А і В |
| 2 | Елементи, які є тільки в масиві А або тільки в масиві В |
| 3 | Елементи, котрі присутні в масиві А, але відсутні в масиві В |
| 4 | Елементи, котрі присутні в обох масивах А і В в декількох екземплярах |
| 5 | Елементи, котрі присутні в декількох *екземплярах* в масиві А, але відсутні в масиві В |
| 6 | Елементи, котрі присутні в декількох екземплярах або тільки в масиві A, або тільки в масиві В |
| 7 | Елементи, котрі присутні в декількох екземплярах або в масиві А, або в масиві В (або в обох масивах) |
| 8 | Елементи масиву А, які повторюються в масиві В декілька раз |
| 9 | Елементи присутні в обох масивах А і В в одному екземплярі |
| 10 | Елементи, присутні в одному екземплярі або тільки в масиві А, або тільки в масиві В |

**Контрольні запитання:**

Обрати алгоритм пошуку та алгоритм сортування та підготувати презентацію з пояснення обраних алгоритмів, їх слабких та сильних сторін.

# Лабораторна робота №9. Структури графів, алгоритми Дейкстри та Флойда

**Мета:** набуття практичних вмінь і навичок з використання алгоритмів Дейкстри та Флойда.

## 9.1. Теоретичний матеріал

Алгоритм Дейкстри - алгоритм на графах, винайдений нідерландським вченим Едсгер Дейкстрой в 1959 році. Знаходить найкоротші шляхи від однієї з вершин графа до всіх інших. Алгоритм працює тільки для графів без ребер негативного ваги. Алгоритм широко застосовується в програмуванні і технологіях, наприклад, його використовують протоколи маршрутизації OSPF і IS-IS.

Кожній вершині з V зіставимо мітку - мінімальне відоме відстань від цієї вершини до a. Алгоритм працює покроково - на кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані.

Ініціалізація. Мітка самої вершини a покладається рівною 0, мітки інших вершин - нескінченності. Це відображає те, що відстані від a до інших вершин поки невідомі. Всі вершини графа позначаються як невідвіданих.

Крок алгоритму. Якщо все вершини відвідані, алгоритм завершується. В іншому випадку, з ще не відвіданих вершин вибирається вершина u, що має мінімальну позначку. Ми розглядаємо різні маршрути, в яких u є передостаннім пунктом. Вершини, в які ведуть ребра з u, назвемо сусідами цієї вершини. Для кожного сусіда вершини u, крім позначених відвідані, розглянемо нову довжину шляху, що дорівнює сумі значень поточної мітки u і довжини ребра, що з'єднує u з цим сусідом. Якщо отримане значення довжини менше значення мітки сусіда, замінимо значення мітки отриманим значенням довжини. Розглянувши всіх сусідів, позначимо вершину u як відвіданих і повторимо крок алгоритму.

Алгоритм Флойда-Воршелла використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребер (але без від'ємнозначних циклів). При звичайній реалізації алгоритм видасть довжини (сумарні ваги) найкоротших шляхів між всіма парами вершин, хоча він не видасть інформацію про самі шляхи. Різні версії алгоритму також використовуються для знаходження транзитивного замикання в відношенні R {\displaystyle R} R, або (враховуючи Метод Шульце), для знаходження найбільшого шляху між всіма парами вершин у зваженому графі.

## 9.2. Завдання для виконання

Написати програму пошуку найкоротшого шляху між заданими вершинами графу за допомогою алгоритму Дейкстри та алгоритму Флойда.

**Варіанти**

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

**Контрольні питання:**

1. Для чого використовуються алгоритми пошуку шляхів на графах?
2. В чому особливість алгоритму Дейкстри.
3. Опишіть і-ту ітерацію алгоритму Дейкстри.
4. В чому особливість алгоритму Флойда.
5. Опишіть і-ту ітерацію алгоритму Флойда.
6. Наведіть порівняльну характеристику алгоритмів Дейкстри та Флойда.
7. Оцініть час, потрібний для виконання алгоритму Дейкстри.
8. Оцініть час, потрібний для виконання алгоритму Флойда.
9. Наведіть приклади доцільності використання алгоритму Дейкстри.
10. Наведіть приклади доцільності використання алгоритму Флойда.

# Лабораторна робота №10. Орієнтовані графи

**Мета:** набуття практичних вмінь і навичок з використання алгоритмів Дейкстри та Флойда.

## 10.1. Теоретичний матеріал

Теоретичний матеріал наведено в попередньому розділі.

## 10.2. Завдання для виконання

Написати програму пошуку найкоротшого шляху між заданими вершинами графу за допомогою алгоритму Дейкстри та алгоритму Флойда.

**Варіанти**

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |

**Контрольні питання:**

1. Для чого використовуються алгоритми пошуку шляхів на графах?
2. В чому особливість алгоритму Дейкстри.
3. Опишіть і-ту ітерацію алгоритму Дейкстри.
4. В чому особливість алгоритму Флойда.
5. Опишіть і-ту ітерацію алгоритму Флойда.
6. Наведіть порівняльну характеристику алгоритмів Дейкстри та Флойда.
7. Оцініть час, потрібний для виконання алгоритму Дейкстри.
8. Оцініть час, потрібний для виконання алгоритму Флойда.
9. Наведіть приклади доцільності використання алгоритму Дейкстри.
10. Наведіть приклади доцільності використання алгоритму Флойда.

# Лабораторна робота №11. Рекурсивні алгоритми

**Мета:** засвоєння теоретичного матеріалу та набуття практичного досвіду створення рекурсивних алгоритмів та написання відповідних їм програм.

## 11.1. Теоретичний матеріал

Рекурсивна функція - це числова функція f (n) {\ displaystyle f (n)} f (n) числового аргументу, яка в своєму записі містить себе ж. Такий запис дозволяє обчислювати значення f (n) {\ displaystyle f (n)} f (n) на основі значень f (n - 1), f (n - 2), ... {\ displaystyle f (n-1), f (n-2), \ ldots} f (n-1), f (n-2), \ ldots, подібно міркування по індукції. Щоб обчислення завершувалося для будь-якого n {\ displaystyle n} n, необхідно, щоб для деяких n {\ displaystyle n} n функція була визначена нерекурсівние (наприклад, для n = 0, 1 {\ displaystyle n = 0,1} n = 0 , 1).

У зв'язку з накладними витратами корисно знати, чи є у рекурсивної функції нерекурсівние (замкнута) форма.

Замкнута форма може бути знайдена не для всіх рекурсивних функцій (співвідношень). Для деяких з них знайдені лише наближені замкнуті форми. Деякі рекурсивні співвідношення, такі як факторіал, вважаються елементарними математичними операціями.

## 11.2. Завдання для виконання

**Завдання**:

Дано натуральне число n. Знайти суму перших n членів ряду чисел, заданого рекурентною формулою.

Розв’язати задачу трьома способами:

1. в програмі використати рекурсивну процедуру або функцію,яка виконує обчислення і членів ряду, і суми на рекурсивному спуску;
2. в програмі використати рекурсивну процедуру або функцію,яка виконує обчислення і членів ряду, і суми на рекурсивному поверненні;
3. в програмі використати рекурсивну процедуру або функцію,яка виконує обчислення членів ряду на рекурсивному спуску, а обчислення сумина рекурсивному поверненні.

**Варіанти**

1. F1 = 1, для ∀ i>1 Fi = Fi–1 + 2 ⋅ sin(i)
2. F1 = 1, для ∀ i>1 Fi = 2 ⋅ Fi–1 – cos(i)
3. F1 = 1, для ∀ i>1 Fi = 2 ⋅ cos(Fi–1) + i
4. F1 = 1, для ∀ i>1 Fi = Fi–1⋅ ln(i) / 8
5. F1 = 1, для ∀ i>1 Fi = sin(Fi–1 + cos(i))
6. F1 = 3, для ∀ i>1 Fi = ln(Fi–1) + i / 3
7. F1 = 1, для ∀ i>1 Fi = Fi–1 + ei / 100
8. F1 = 1, для ∀ i>1 Fi = ln(Fi–1⋅ i) + 9
9. F1 = 1, для ∀ i>1 Fi = cos(Fi–1) ⋅ 100 / i
10. F1 = 1, для ∀ i>1 Fi = 3 ⋅ sin(Fi–1) + ln(i)

**Контрольні питання**

1. Визначення рекурсивного об’єкту.
2. Визначення глибини та поточного рівня рекурсії.
3. Форма виконання рекурсивних дій на рекурсивному спуску.
4. Форма виконання рекурсивних дій на рекурсивному поверненні.
5. Форма виконання рекурсивних дій на як рекурсивному спуску, так і на рекурсивному поверненні.

# Лабораторна робота №12. Незв’язні динамічні структури

**Мета:** засвоєння теоретичного матеріалу та набуття практичного досвіду використання незв’язаних динамічних структур даних при складанні різних алгоритмів.

## 12.1. Теоретичний матеріал

Теоретичний матеріал подано у відповідній лекції.

## 12.2. Завдання для виконання

Написати програму реалізуючі масиви як незв’зяні динамічні структури.

**Варіанти**

1. Дано три масиви A[10], B[12], C[8] цілого типу. При послідовному перегляді знайти в кожному масиві добуток стількох елементів, сума яких не перевищує заданого цілого М.
2. Дано дві матриці A[6,6], B[8,8]довгого цілого типу. В кожній з матриць знайти суму елементів над головною діагоналлю і добуток елементів під нею.
3. Дано дві матриці A[5,5], B[7,7] натурального типу. Присвоїти змінній PA значення 1, якщо елементи матриці А симетричні відносно головної діагоналі, і значення 0 – в іншому випадку.Аналогічно присвоїти значення змінній PВ для матриці В.
4. Дано дві матриці A[5,6], B[7,4] цілого типу. Для кожної з матриць підрахувати кількість елементів, які більші за задане ціле число k і мають непарну суму координат.
5. Дано три масиви A[9], B[11], C[14]дійсного типу. В кожному з них знайти суму елементів, розміщених після першого від’ємного елемента, і суму елементів, розміщених до нього.
6. Дано три матриці A[3,9], B[6,5], C[5,7] байтового типу. Обчислити суму елементів в кожному рядку заданих матриць.
7. Дано три масиви X[10], Y[15], Z[12] дійсного типу. В кожному з цих масивів елементи з номерами, кратними трьом, заміни-ти напів сумою двох попередніх елементів, якщо їх добуток менше нуля, і залишити без змін в іншому випадку.
8. Дано три масиви A[19], B[17], C[23] цілого типу. Для кожного масиву визначити кількість додатних елементів на парних позиціях і суму елементів на непарних позиціях.
9. Дано три масиви A[12], B[15], C[20] натурального типу. Для кожного масиву визначити добуток елементів на непарних позиціях та знайти перший максимальний елемент масиву.
10. Дано дві матриці A[4,5], B[5,8] цілого типу. При обході порядках знайти останній мінімальний елемент кожної матриці, а також суму елементів матриці.

**Контрольні питання**

1. Яких типів допускаються динамічні дані?
2. В чому полягає відмінність оголошення звичайних змінних і незв’язаних динамічних змінних такого самого типу?
3. Що виконує операція розйменування динамічної змінної?
4. За допомогою яких процедур чи функцій виконується виділення пам’яті для динамічних даних?
5. В яких станах може знаходитись динамічна змінна?
6. В яких випадках динамічна змінна може знаходитись у невизначеному стані?
7. В чому полягає відмінність стану динамічної змінної nil та її невизначеного стану?
8. Які значення будуть порівнюватися, якщо виконується операція порівняння над ідентифікаторами динамічних змінних?
9. В чому полягають динамічні властивості незв’язаних динамічних даних?
10. В яких випадках доцільно використовувати незв’язані динамічні дані?

# Рекомендовані джерела

**Основна література**

1. Bäckman K. Structured Programming with C++ / Kjell Bäckman. – Frederiksberg: Ventus Publishing ApS, 2012. – 246 с.
2. Shaffer C. Data Structures and Algorithm Analysis / Clifford A. Shaffer. – Blacksburg: Department of Computer Science Virginia Tech, 2013. – 615 с.
3. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона / Н.Вирт. - М.: ДМК Пресс, 2010. - 272с.
4. Гамма, Р.Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.
5. Конова Е. А., Поллак Г. А.Конова Е. А., Поллак Г. А. Алгоритмы и программы. Язык С++: Учебное пособие. - 2-е изд., стер. - СПб.: Издательство «Лань», 2017. - 384 с.: ил. - (Учебники для вузов. Специальная литература).
6. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; пер. с англ. под ред. А. Шеня. - 3-е изд. М. : ООО «И. Д. Вильямс», 2013. - 1398 с.
7. Прата С. Язык программирования C++. Лекции и упражнения / Стивен Прата. – Москва: Вильямс, 2017. – 1248 с.

**Додаткова література**

1. Brokken F. C++ Annotations Version 11.1.0 [Електронний ресурс] / Frank B. Brokken // Center of Information Technology, University of Groningen. – 2019. – Режим доступу до ресурсу: http://www.icce.rug.nl/documents/cplusplus/
2. Halterman R. Fundamentals of Programming C++ / Richard L. Halterman. – Collegedale: Southern Adventist University, 2019. – 766 с.
3. Бьерн Страуструп. Язык программирования С++. – Москва: Бином, 2015. – 1136 с.
4. Джейсон Мейнджер Java: Основы программирования .- McGraw-Hill,Inc.,1996, Издательская группа BHV, Киев,1997
5. Джо Вебер Технология Java в подлиннике .- QUE Corporation, 1996, "BHV-Санкт-Петербург",1997
6. Доусон М. Изучаем С++ через программирование игр / Майкл Доусон. – Санкт-Петербург: Питер СПб, 2016. – 352 с.
7. Липпман С. Язык программирования C++. Базовый курс / С. Липпман, Ж. Лажойе, Б. Му. – Москва: Вильямс, 2014. – 1120 с.
8. Майкл Эферган Java: справочник .- QUE Corporation, 1997, Издательство "Питер Ком", 1998
9. Чен М.С. и др. Программирование на JAVA:1001 совет:Наиболее полное руководство по Java и Visual J++ :Пер.с англ./Чен М.С.,Грифис С.В.,Изи Э.Ф..-Минск:Попурри,1997.-640с.ил.

Навчальне видання

Плескач Валентина Леонідівна

Пирог Микола Володимирович

**Методичні вказівки до виконання лабораторних робіт**

**з курсу «Алгоритми і структури даних»**

для студентів, які навчаються за спеціальністю

122 «Комп’ютерні науки»,

освітньої програми «Прикладне програмування»