Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

Інесса Краснокутська

АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ

Лабораторний практикум

для студентів спеціальності
8.04030101 «Прикладна математика»
денної та заочної форм навчання

Чернівці
Чернівецький національний університет
2017

ББК

УДК 519.683, 378.147.88

К

Інесса Краснокутська

Алгоритми та структури даних: Лабораторний практикум. — Чернівці: Чернівецький національний університет, 2017. — 48 с.

Кафедра прикладної математики та інформаційних технологій

Затверджено
на засіданні кафедри прикладної
математики та інформаційних
технологій як лабораторний
практикум з дисципліни
«Алгоритми та структури даних»
Протокол №1 від 30.08.2017

ЗМІСТ

ЛАБОРАТОРНІ РОБОТИ

МОДУЛЬ 1

Алгебра складності 4
Оцінка складності алгоритму 7
Однонаправлені та двонаправлені списки 9
Використання стеку для обчислення виразу, записаного в оберненій польській нотації 12
Стеки та черги 13
Кільцеві списки 16

МОДУЛЬ 2

Методи розробки алгоритмів 18
Розділяй та володарюй 22
Алгоритми сортування 23
Бінарні дерева та бінарні дерева пошуку 26
Хеш-таблиці 28

ЗАВДАННЯ ДЛЯ ЛІТНЬОЇ ОБЧИСЛЮВАЛЬНОЇ ПРАКТИКИ

Експериментальне дослідження ефективності реалізацій асоціативного масиву за допомогою бінарного дерева пошуку, хеш-таблиці з ланцюжками та хеш-таблиці з відкритою адресацією 31
Робота з абстрактними классами 38

Лабораторна робота №1.1

Алгебра складності

1) Покажіть, що $f(n)=3.7n^2+100.8n+10^6$ має порядок $O(n^2)$ при $n\to\infty$.
2) Покажіть, що $f(n)=\frac{2^n}{100}-100$ має порядок $O(2^n)$ при $n\to\infty$.
3) Покажіть, що наступне твердження вірне: *17* має порядок $O(1)$.
4) Покажіть, що наступне твердження вірне: $\frac{n(n-1)}{2}$ має порядок $O(n^2)$.
5) Визначте, яке з тверджень a) чи b) вірне для $f(n)=\frac{n^2-n}{2}$, $g(n)=6n$:
   a) $f(n)$ є $O(g(n))$,
   b) $g(n)$ є $O(f(n))$.
6) Визначте, яке з тверджень a) чи b) вірне для $f(n)=n+2\sqrt{n}$, $g(n)=n^2$:
   a) $f(n)$ є $O(g(n))$,
   b) $g(n)$ є $O(f(n))$.
7) Визначте, яке з тверджень a) чи b) вірне для $f(n)=n^2+3n+4$, $g(n)=n^3$:
   a) $f(n)$ є $O(g(n))$,
   b) $g(n)$ є $O(f(n))$.
8) Покажіть, що поліном $f(n)=2n^5+6n^4+6n^2+18$ є $O(n^5)$. Чи є він $o(n^{5.1})$, $o(e^n)$, $o(2^n)$?
9) Покажіть, що функція $f(n)=1000\sqrt{n}$ є $o(n^5)$.
10) Визначте порядок росту для функції $T(n)=(n+1)^2$, $T(1)=4$, $T(0)=1$. При яких значеннях сталих n_0 і C функція $T(n)$ буде мати порядок $O(n^2)$? Чи можна за n_0 взяти *0*?

11) При яких значеннях сталих n_0 і C функція $T(n) = 3n^3 + 2n^2$ буде мати порядок $O(n^3)$?

12) Чи можна сказати, що функція $T(n) = 3n^3 + 2n^2$ має порядок $O(n^4)$? Яке з тверджень a) чи b) більш слабке?

    a) $T(n) \in O(n^4)$,

    b) $T(n) \in O(n^3)$.

13) Покажіть, що функція $f(n) = 3^n$ не може мати порядок $O(2^n)$.

14) Покажіть, що функція $f(n) = n^5$ не є $O(n^4)$.

15) Перевірте, що $T(n) = n^3 + 2n^2 \in \Omega(n^3)$.

16) Перевірте, що $T(n) = \begin{cases} n, & \text{для непарних } n \geq 1, \\ \dfrac{n^2}{100}, & \text{для парних } n \geq 0 \end{cases} \in \Omega(n^2)$.

17) Нехай для певного комп'ютера і компілятора одна програма виконується за $100n^2$ мілісекунд, а інша — за $5n^3$ мілісекунд. Чи може друга програма мати перевагу по швидкодії за першу?

18) Для чотирьох програм різної складності $2^n, \dfrac{n^3}{2}, 5n^2, 100n$ визначте, який максимальний розмір задачі, яку можна розв'язати за 1000 секунд? За 10^4 секунд? Якому алгоритму надати перевагу?

19) Чи вірно, що програми з часом виконання $O(n)$ мають таку перевагу: 10-кратне збільшення розміру задачі при 10-кратному підвищення продуктивності ПК?

20) Чи вірно, що збільшення швидкості ПК у 10 разів призведе до збільшення тільки на 30% розміру задачі при використанні програми з часом виконання $O(2^n)$?

21) Чи вірно, що незалежно від швидкодії ПК, програма з часом виконання $O(2^n)$ може розв'язувати тільки дуже невеликі задачі.

22) Доведіть правило сум для двох фрагментів програми. Нехай $T_1(n)$ є $O(f(n))$, $T_2(n)$ є $O(g(n))$ — час виконання двох програмних фрагментів P_1 і P_2 відповідно. Визначте час послідовного виконання фрагментів P_1 і P_2.

23) Знайдіть час послідовного виконання трьох фрагментів програми з порядками $O(n^2)$, $O(n^3)$, $O(n \log n)$.

24) Доведіть правило добутків для двох фрагментів програми. Нехай $T_1(n)$ є $O(f(n))$, $T_2(n)$ є $O(g(n))$ — час виконання двох програмних фрагментів P_1 і P_2 відповідно.

25) Покажіть, що $O(Cf(n))$ еквівалентно $O(f(n))$, де стала $C > 0$.

26) Чи $O\left(\frac{n^2}{2}\right)$ еквівалентно $O(n^2)$?

27) Покажіть, що наступне твердження вірне:
$\max(n^3, 10n^2)$ має порядок $O(n^2)$.

28) Покажіть, що якщо $p(n)$ — поліном k-го порядку з додатнім старшим коефіцієнтом, то $p(n)$ є $O(n^k)$ і $p(n)$ є $\Omega(n^k)$.

29) Припустимо, що $T_1(n)$ є $\Omega(f(n))$, $T_2(n)$ є $\Omega(g(n))$. Яке з наступних тверджень вірне?

a) $T_1(n) + T_2(n)$ є $O(\max(f(n), g(n)))$,

b) $T_1(n)T_2(n)$ є $O(f(n)g(n))$.

30) Розташуйте наступні функції в порядку степені росту: n, $\sqrt{n}$, $\log n$, $\log\log n$, $\log^2 n$, $n/\log n$, $\sqrt{n}\log^2 n$, $\left(\frac{1}{3}\right)^n$, $\left(\frac{3}{2}\right)^n$, 17.

31) Чи можна стверджувати, що 2^{n+1} є $O(2^n)$? 2^{n+1} є $o(2^n)$?

32) Чи можна стверджувати, що 2^{2n} є $O(2^n)$?

33) Для кожної пари нижченаведених функцій f і g виконується тільки одна з рівностей $f = O(g)$ або $g = O(f)$. Визначити, який із випадків має місце

a) $f(n) = n + n\log_2 n,\ g(n) = n\sqrt{n},$

b) $f(n) = n\log_2 n,\ g(n) = \frac{n\sqrt{n}}{2},$

c) $f(n) = 2\log_2 n^2,\ g(n) = \log_2 n + 1.$

34) Чи $f(n) = 2^n$ є $o(n!)$, $O(2^{n+1})$, $o(5^{n-1})$?

35) Чи $f(n) = 1000\sqrt{n}$ є $o(n)$?

36) Покажіть, що *n!* має порядок $O(n^n)$ при $n \to \infty$.

37) Покажіть, що 2^n росте при $n \to \infty$ швидше, ніж будь-який поліном від n скінченного порядку.

38) Чи алгоритм, що розв'язує задачу розміру *n* за $O(2^n)$ кроків, ефективніший за алгоритм, який розв'язує її за $O(n!)$ чи $O(n^n)$ кроків?

Лабораторна робота №1.2

Оцінка складності алгоритму

Розробіть алгоритм для реалізації поставленої задачі. Оцініть його складність за вхідними даними.

1) Перевести довільне додатне ціле число, записане арабськими цифрами в систему римських цифр.
2) Перевести довільне додатне ціле число, записане римськими цифрами в систему арабських цифр.
3) Для довільної трійки чисел визначте чи існує трикутник з такими сторонами.

4) Обчисліть кількість знаків, що міститься в десятковому запису числа *n!*
5) Розробіть алгоритм, що визначатиме, скільки часу буде потрібно для виконання 2^n, n^n і *n!* елементарних операцій для заданого *n*.
6) Розробіть алгоритм для обчислення $G(m,n)=k$, де k обчислюється таким чином: $n^{k-1} \le m! \le n^k$.
7) Реалізуйте алгоритм знаходження максимуму в масиві значень. Користуючись генератором випадкових чисел, для *n=20* створіть вхідний масив. Побудуйте усереднений профіль для Вашої програми за 10 вхідними наборами. Профіль програми для певного вводу — це гістограма, що показує, скільки разів виконується кожен оператор програми під час її роботи.
8) Обчисліть найбільший спільний дільник двох чисел.
9) Обчисліть найменше спільне кратне двох чисел.
10) За допомогою решета Ератосфена виведіть прості числа, які не перевищують *n*.
11) Перетворіть десяткове число в ланцюговий дріб.
12) Перетворіть ланцюговий дріб в десяткове число.
13) Розкладіть число на прості множники.
14) Реалізуйте ділення цілих чисел з довільною точністю.
15) Обчисліть визначник заданої матриці.
16) Реалізуйте метод Крамера для розв'язування СЛАР.
17) Реалізуйте метод Гауса для розв'язування СЛАР.
18) Задано число $1 \le n \le 10^7$. Необхідно знайти останню цифру *n*-го числа Фібоначчі.
19) Задані цілі числа $1 \le n \le 10^8$ і $2 \le m \le 10^5$. Необхідно знайти остачу від ділення *n*-го числа Фібоначчі на *m*.

Лабораторна робота №2.1

Однонаправлені та двонаправлені списки

В кожному завданні передбачити можливіть заповнення структури даних випадковими значеннями або тими, що вводить користувач.

1) Створити динамічний список елементами якого є цілі числа. Створити функцію **DoubleOdd**, яка дублює в списку всі непарні елементи (тобто за кожним непарним елементом має йти ще один рівний йому).
2) Створити динамічний список, елементами якого є цілі числа. Реалізувати функцію **IsUnique** яка повертає **1** якщо в списку немає елементів що повторюються і **0** в протилежному випадку.
3) Створити динамічний список, елементами якого є цілі числа. Реалізувати функцію **Concatenate** яка повертає новий список який є об’єднанням двох списків, які передаються в функцію як параметри (після роботи функції ці списки мають залишитись без змін).
4) Створити динамічний список елементами якого є символи. Створити функцію **EraseChars**, яка вилучає з списку якій передається як перший параметр всі символи які входять в список, що передається як другий параметр.
5) Створити динамічний список елементами якого є цілі числа. Створити функцію **Revert**, яка переставляє елементи списку в зворотному порядку.
6) Створити динамічний список елементами якого є символи. Створити функцію **CheckBrackets**, яка повертає **1** якщо дужки в списку є збалансованими (кожній закриваючій дужці передує відкриваюча) і **0** в протилежному випадку.

7) Створити динамічний список елементами якого є символи. Реалізувати функцію **RevertCase**, яка змінює в списку маленькі латинські літери на великі і навпаки великі латинські літери на маленькі.
8) Створити динамічний список елементами якого є десяткові цифри. Список містить запис числа в десяткові системі числення зліва-направо в порядку зменьшеня розряду цифри. Написати функцію Sum яка повертає список який є сумою двох інших списків, які передаються як параметри.
9) Створити динамічний список елементами якого є символи. Реалізувати функцію **LettersCount**, яка повертає кількість різних маленьких латинських літер в списку.
10) Створити динамічний список елементами якого є символи. Реалізувати функцію **BigLetters** яка повертає новий список, який складається з великих латинських літер, що не містяться в списку який передається як параметр.
11) Створити динамічний список якій визначає многочлен, для цього кожен елемент списку повинен містити ціле число – степінь **X** та дійсне число – коефіцієнт при **X**. Створити функцію **GetValue** яка для заданого **X** повертає значення многочлена.
12) Створити динамічний список якій визначає многочлен, для цього кожен елемент списку повинен містити ціле число – степінь **X** та дійсне число – коефіцієнт при **X**. Створити функцію **Simplify** яка зводить в списку доданки при однакових степенях **X**.
13) Створити динамічний список якій визначає многочлен, для цього кожен елемент списку повинен містити ціле число – степінь **X** та дійсне число – коефіцієнт при **X**. Створити функцію **Multiply**, яка створює новий список який визначає многочлен який є добутком двох многочленів, що передаються як параметри.

14) Створити динамічний список елементами якого є цілі числа. Створити функцію **Union**, яка будує об'єднання двох списків, що передаються як параметри (після роботи функції ці списки мають залишитись без змін). Під об'єднанням списків **A** та **B** тут розуміється список якій складається з елементів які входять або в **A** або в **B** взятих по одному разу.
15) Створити динамічний список елементами якого є цілі числа. Створити функцію **Intersection**, яка будує перетин двох списків, що передаються як параметри (після роботи функції ці списки мають залишитись без змін). Під перетином списків **A** та **B** тут розуміється список якій складається з елементів які входять одночасно і в **A** і в **B** взятих по одному разу.
16) Створити динамічний список якій визначає многочлен, для цього кожен елемент списку повинен містити ціле число – степінь **X** та дійсне число – коефіцієнт при **X**. Створити функцію **Derivative** яка повертає список який визначає похідну многочлена який задається як параметр.
17) Створити динамічний список елементами якого є цілі числа. Створити функцію **DelChains**, яка видаляє з списку всі ланцюжки однакових елементів які йдуть підряд довжиною більше трьох.
18) Створити динамічний список елементами якого є цілі числа. Створити функцію **Function**, яка знищує зі списку всі від'ємні елементи, а додатні замінює на їх квадрати.

Лабораторна робота №2.2

Використання стеку для обчислення виразу, записаного в оберненій польській нотації

Обчислити значення виразу. Оформити звіт в зошиті.

1) 1 2 / 2 3 + 9 2 - sin 2 ^ 6 7 / - / +
2) 1 3 2 * + 2 3 + 9 2 - 8 * 13 2 / - / +
3) 1 3 exp 2 * - 2 3 4 * + + 18 2 - cos +
4) 132 118 2 * - 2 7 / cos + 11 2 ^ -
5) 2 12 – 23 + 2 1 + 3 8 cos - / +
6) 132 54 - 41 18 - 2 1 - * + 114 sin +
7) 149 54 12 13 - * - 3 2 - cos 2 / +
8) 14 15 * 48 / 11 - 12 2 3 2 / - cos * +
9) 144 1000 - 52 + 48 - 62 11 0 cos - * 2 / *
10) 5 2 / 2 3 + 9 2 – sin 2 ^ 6 7 / - / *
11) 6 3 2 * + 2 3 + 9 2 - 8 * 13 2 / - / -
12) 7 3 exp 2 * - 2 3 4 * + + 18 2 - cos -
13) 832 118 2 * - 2 7 / cos + 11 2 ^ -
14) 9 12 - 23 + 2 1 + 3 8 cos - / -
15) 232 54 - 41 18 - 2 1 - * + 114 sin -
16) 349 54 12 13 - * - 3 2 - cos 2 / -
17) 44 15 * 48 / 11 - 12 2 3 2 / - cos * -
18) 544 1000 - 52 + 48 - 62 11 0 cos - * 2 / -
19) 6 2 / 2 3 + 9 2 - sin 2 ^ 6 7 / - / -
20) 7 3 2 * + 2 3 + 9 2 - 8 * 13 2 / - / -
21) 39 54 2 13 + * - 3 2 - cos 2 / -
22) 4 1 - 48 / 11 - 12 2 3 2 / - cos * -
23) 544 2 - 52 - 48 - 62 11 0 sin - * 2 / /
24) 6 2 / 13 3 + 9 2 - cos 2 ^ 6 7 / - / *
25) 71 3 2 * + 2 13 + 9 2 - 8 * 13 2 / - / +

Лабораторна робота №2.3

Стеки та черги

В кожному завданні передбачити можливіть заповнення структури даних випадковими значеннями або тими, що вводить користувач.

1)
   a) Написати функцію, яка обчислює кількість елементів стеку, у яких рівні «сусіди» (наступний та попередній елементи).
   b) Написати функцію, яка з однієї черги **Queue** будує дві нових: **Queue1** з додатніх елементів і **Queue2** – з решти елементів черги.

2)
   a) Написати функцію, яка знаходить мінімальний елемент в стекові і поміщує його у вершину стеку.
   c) Написати функцію, яка формує чергу **Queue**, включивши в нею по одному разу елементи, які входять в одну із черг **Queue1** та **Queue2**.

3)
   a) Написати функцію, яка визначає чи є в стекові хоча б один елемент, що співпадає з наступним за ним елементом.
   b) Написати функцію, необхідну для визначення середнього арифметичного елементів черги між першим і останнім входженням елемента **E**, якщо елемент **E** входить в чергу не менше двох разів.

4)
   a) Написати функцію, яка формує стек **St**, включивши в нього по одному разу елементи, що входять хоча б в один із стеків **St1** та **St2**.
   b) Написати функцію, необхідну для визначення мінімального елементу черги.

5)
   a) Написати функцію, яка визначає чи є в стекові **St1** елементи, які не входять в стек **St2**.
   b) Написати функцію, яка в черзі визначає кількість елементів, у яких однакові сусіди (наступний і попередній елементи).

6)

a) Написати функцію, яка знаходить максимальний елемент в стекові.

d) Написати функцію, яка форму чергу **Queue**, включивши в неї по одному разу елементи, що входять в хоча б одну з черг **Queue1** або **Queue2**.

7)

a) Написати функцію, яка знаходить мінімальний елемент в стекові.

b) Написати функцію, яка форму чергу **Queue**, включивши в неї по одному разу елементи, що входять в хоча б одну з черг **Queue1** або **Queue2**.

8)

a) Написати функцію, необхідну для визначення середнього арифметичного елементів стеку між першим і останнім входженням елемента **E**, якщо елемент **E** входить в чергу не менше двох разів.

b) Написати функцію, яка перевіряє рівність двох черг.

9)

a) Написати функцію, що необхідна для реалізації розбиття одного стеку на два: в перший занести елементи, менші за введений з клавіатури **X**, в інший – більші.

b) Написати функцію, яка визначає чи є в черзі хоча б один елемент, що співпадає з наступним за ним елементом.

10)

a) Написати функцію, що необхідна для реалізації об’єднання двох стеків в один.

b) Написати функцію, яка перевіряє чи є в чергах **Queue1** та **Queue2** однакові елементи.

11)

a) Написати функцію, що перемістить в вершину стеку елемент, який був першим доданим в стек.

b) Написати функцію, яка необхідна для об’єднання двох впорядкованих черг в одну чергу, щоб впорядкованість не порушувалась.

12)

a) Написати функцію, яка підраховує кількість входжень елемента **E** в стек.

b) Написати функцію, що обчислює середнє арифметичне всіх елементів черги.

13)

a) Написати функцію, що обчислює середнє арифметичне всіх елементів стеку.

b) Написати функцію, що підраховує кількість входжень елемента **E** в чергу.

14)

a) Написати функцію, що підраховує кількість входжень елемента **E** в стек.

b) Написати функцію, що формує нову чергу з тих елементів черги, в яких немає однакових «сусідів» (наступний та попередній елементи).

15)

a) Написати функцію, що видаляє з стеку ті елементи, які є квадратами чисел.

b) Написати функцію, що формує нову чергу з тих елементів черги, які є простими числами.

16)

a) Написати функцію, що знищує з стеку елементи, що повторюються залишивши по одному.

b) Написати функцію, що поміщує в початок черги парні, а в кінець – непарні елементи.

17)

a) Написати функцію, що видаляє зі стека всі числа, які не повторюються.

b) Написати функцію, що видаляє з черги числа, кратні заданому користувачем числу.

18)

a) Написати функцію створення стеку з цілих чисел, в якому кожен елемент дорівнює сумі попередніх елементів. Перший елемент дорівнює одиниці.

b) Написати функцію, що записує чергу в зворотному порядку.

Лабораторна робота №2.4

Кільцеві списки

В кожному завданні передбачити можливість заповнення структури даних випадковими значення або тими, що вводить користувач.

1) Дано кільцевий список, що містить 22 прізвища гравців футбольної команди. Розбити гравців на 2 групи по 11 чоловік. У другу групу потрапляє кожен 12-й гравець.
2) Дано два кільцевих списки, що містять прізвища спортсменів двох фехтувальних команд. Провести жеребкування. У першій команді вибирається кожен *N*-й гравець, а в другій - кожен *M*-й.
3) Дано список. Створити функцію **SmartErase** яка циклічно рухається по списку вилучаючи кожний третій елемент, поки в списку не залишиться два елементи.
4) Дано два кільцевих списки, що містять прізвища учасників лотереї та найменування призів. Виграє *N* чоловік (кожен *K*-й). Число для перерахунку призів – *T*. Вивести результати лотереї.
5) Дано два списки, що містять прізвища учнів та номери екзаменаційних білетів. Число перерахунку для білетів – *E*, для учнів – *K*. Визначити номери білетів, що будуть витягнені учнями
6) Дано два списки, що містять прізвища студентів двох груп. Перевести *L* студентів з першої групи в другу. Число перерахунку – *K*.
7) Дано два списки, що містять перелік товарів, що виробляються концерном **BOSH** та **PHILIPS**. Створити список товарів, що випускаються як однієї так і іншою фірмою.
8) Дано два списки, що містять перелік товарів і прізвища покупців. Кожен *N*-й покупець купує *M*-й товар. Вивести список покупок.

9) Скласти програму, що в кільцевій список з *N* елементів додає m нових елементів так, щоб новий елемент вставлявся через *K* елементів.
10) Текст, що закінчується крапкою, занести в динамічний кільцевий список. Вивести на екран текст з кільцевого списку *N* раз. *N* ввести з клавіатури.
11) Дано кільцевий список. Описати функцію, що підраховує кількість елементів у списку у яких рівні «сусіди».
12) Дано лінійний односпрямований список. Побудувати два кільцевих списки, що містять: перший лише парні значення, другий лише непарні.
13) Дано кільцевий список. Описати функцію, що підраховує середнє арифметичне елементів у списку у яких лівий «сусід» більше «правого».
14) З введених слів сформувати кільцевий однозв’язний список. Створити новий список, організований як стек, в який переписати всі слова, що складаються з тих самих літер, що й перше введене слово, а з вихідного списку їх видалити. Надрукувати модифікований базовий список та новостворений список або вивести повідомлення про відсутність шуканих слів. Використати функцію, яка перевіряє, чи два задані символьні рядки складаються з однакових символів.
15) З введених довгих цілих чисел сформувати двозв’язний кільцевий список (у списку зберігати тільки адреси чисел, які розташувати в динамічній пам’яті окремо). Потім вилучити зі списку всі елементи, які більше, ніж у три рази перевищують значення найменшого елемента списку. Розробити окрему функцію для визначення найменшого елемента списку.

Лабораторна робота №3.1

Методи розробки алгоритмів

Головоломки

1) *«Кросворд»* Напишіть програму складання кросвордів. Припускається, що вихідними даними є конфігурація *N*M* (деяке розміщення порожніх та заповнених квадратів) та список слів, що складаються з *max(N, M)* букв або менше. Результатом має бути розміщення цих слів, що утворює кросворд або повідомлення про те, що така конфігурація неможлива. Слова зчитувати з досить великого текстового файлу (художньої книжки). Вивести кількість слів, які довелось переглянути в книжці, щоб утворити кросворд. Записати кросворд у файл. Виконати роботу візуально, а не в консолі.
2) *«Перестановки в стеці»* Нехай цілі числа *1, 2, ..., N* послідовно приходять на вхід стеку. Розглянувши всі можливі послідовності операцій **push**() і **pop**() визначити, які з *N!* всеможливих перестановок можна отримати на виході. Вивести їх, пронумерувавши. Надати користувачу можливість вибравши номер перестановки побачити коментар до послідовності дій, наприклад для перестановки *2314* таким коментарем буде **push(1)**, **push(2)**, **pop(2)**, **push(3)**, **pop(3)**, **pop(1)**, **push(4)**, **pop(4)**. Продемонструвати візуально ці дії. Записати всі перестановки, які задовольняють умову задачі у файл. Виконати роботу візуально, а не в консолі.
3) *«Перестановки в деці»* Виконати задачу «перестановки в стеці» для деку.
4) *«Обчислення π»* Написати програму обчислення наближеного значення *π = 3.14159...* Наступні співвідношення описують четвертину круга радіусу 1, що знаходиться в першому квадранті двовимірної

декартової системи координат: $x^2 + y^2 \le 1$, $x \ge 0$, $y \ge 0$. Розглянемо одиничний квадрат $0 \le x \le 1$, $0 \le y \le 1$. Виберемо в квадраті *N* випадкових точок. Якщо *n* з них знаходяться в четвертині круга, то відношення *n / N* має апроксимувати площу, тобто $n / N \approx \pi / 4$. Звідси можна приблизно обчислити π. Виконати обчислення для *N = 100, 500, 1000, 5000, 50000*. Продемонструвати роботу програми візуально.

Пошук з поверненням (Backtracking)

5) Знайти розв'язок наступної головоломки. Є вхідна конфігурація (генерується випадково або користувач формує її). Кожен квадрат є рухомим, його можна пересунути на порожню комірку. Мета: отримати кожну з результуючих конфігурацій. Створити ієрархію папок з текстовими файлами, що містять матрицю-конфігурацію на кожному кроці. Зобразити візуально за допомогою анімації.

2	3	X
1	8	4
7	6	5

1	2	3
8	X	4
7	6	5

1	2	3
4	5	6
7	8	X

6) *«Задача про підбір паролю»* Продемонструвати візуально підбір паролю методом пошуку з поверненням. Анімаційно підсвічувати кожен крок алгоритму. Сірим кольором відобразити гілки, які не задовольняють умову, що *N/2* перемикачів мають бути включені. Передбачити можливість задання мінімальної кількості перемикачів, які одночасно мають бути увімкнені. Вивести послідовність вершин, що переглядаються у файл.

7) *«Задача про вихід з лабіринту»* Продемонструвати візуально пошук виходу з лабіринту. Передбачити випадки чотиризв'язної та восьмизв'язної областей. Анімайійно підсвічувати кожен крок алгоритму. Користувач може задати розмір лабіринту *(N*M)* та

стартову позицію. Лабіринт генерувати випадковим чином, або зчитувати з файлу або надати можливість користувачу побудувати його, клікаючи на комірках сітки. Вивести послідовність комірок, що переглядаються в файл.

8) *«Хрестики-нулики»* Задана таблиця *3*3* із символів **X** та **O**, що відмічають ходи відповідно хрестиків та нуликів. Символом # відмічені вільні поля. Таблиця містить рівно три хрестика та три нулика. Гарантується, що партія не завершена, а саме, ніякий рядок, стовпчик чи діагональ не заповнені цілком ні хрестиками, ні нуликами. Визначити, яка сторона виграє при найкращій грі кожного гравця. Продемонструвати ходи візуально.

Алгоритми на шаховій дошці

9) Знайти кількість способів якими можна розкласти *k (k<=N)* слонів одного кольору на шаховій дошці *N*N (N<=12).* (Динамічне програмування.)
10) Вивести максимальну кількість ферзів, що можна розкласти на шаховій дошці *N*N (N<=12)* таким чином, щоб вони не били одне одного.
11) Розкласти на дошці *N*N (N<=12) N* ферзів таким чином, щоб найбільша кількість її клітинок знаходилась поза сутичкою ферзів. Виведіть цю кількість на екран.
12) Визначити мінімальну кількість тур, які можна розкласти на шаховій дошці *N*N (N<=12)* таким чином, щоб ніякі дві тури не загрожували одна одній?
13) Визначити мінімальну кількість тур, які можна розкласти на шаховій дошці *N*N (N<=12)* таким чином, щоб усі тури тримали під загрозою усі поля?

14) Задана шахова дошка *N*M*. Чи існує обхід конем всіх клітинок дошки по одному разу? (Алгоритми з поверненням.)
15) В лівому верхньому куті шахової дошки *N*M* стоїть кінь. Підрахуйте кількість можливих маршрутів коня з верхнього лівого кута в правий нижній.
16) В лівому верхньому куті шахової дошки стоїть король, який може ходити тільки на одну комірку вбік або по діагоналі. За заданими числами *N* та *M* підрахуйте кількість можливих маршрутів короля з верхнього лівого кута в правий нижній.
17) Яку мінімальну кількість слонів можна розкласти на шаховій дошці *N*N (N<=12)* таким чином, щоб вони тримали під загрозою усі поля?
18) Поле шахової дошки визначається парою натуральних чисел, перше з яких задає номер по вертикалі, а друге – номер по горизонталі. Дано натуральні числа *k, l, m, n*. Чи може ферзь за один хід перейти з поля *(k,l)* на поле *(m,n)*? Якщо ні, то визначити як це зробити за два ходи.
19) Дошка складається з *N*M* клітин, розфарбованих в шаховому порядку. Відомо, що клітинка в лівому нижньому куті – чорна. Визначити скільки всього на дошці чорних клітин.
20) Поле шахової дошки визначається парою натуральних чисел, перше з яких задає номер по вертикалі, а друге – номер по горизонталі. Дано натуральні числа *k, l, m, n*. Чи є поля *(k,l)* та *(m,n)* полями одного кольору?
21) За яку найменшу кількість ходів ферзь може обійти всі поля дошки *N*N (N<=12)*?
22) Задані координати двох клітин шахової дошки *(x1, y1)* та *(x2, y2)*. Визначити, чи може слон за один хід перейти з клітинки *(x1, y1)* в клітинку *(x2, y2)* (вивести на екран **Yes** або **No**).

Лабораторна робота №3.2

Розділяй та володарюй

Розробіть алгоритм для реалізації поставленої задачі. Оцініть його складність за вхідними даними.

1) Реалізувати розв’язання головоломки «Ханойські вежі».
2) Розробіть розклад ігор спортивних команд.
3) Реалізуйте алгоритм множення n-розрядних чисел.
4) Реалізуйте алгоритм швидкого множення матриць (алгоритм Штрассена).
5) Реалізуйте алгоритм злиття *k* впорядкованих масивів.
6) Реалізуйте множення многочленів, використовуючи швидке перетворення Фур’є.
7) Знайдіть випуклу оболонку заданого масиву точок.
8) Реалізуйте задачу про пошук пари найближчих точок.
9) Знищіть з масиву усі дублікати.
10) Знайдіть найбільшу спільну підпослідовність в двох символьних масивах.
11) Реалізуйте метод ділення навпіл для знаходження розв’язків нелінійних рівнянь.
12) Підрахуйте кількість інверсій в масиві.
13) Знайдіть медіану масиву.
14) Знайти представника більшості в масиві.
15) Реалізуйте алгоритм злиття двох масивів в один.
16) Реалізуйте алгоритм швидкого пошуку.
17) Дослідіть чи існує в масиві такий індекс і, що $a_i=i$.
18) Реалізувати алгоритм знаходження найбільшого спільного дільника.
19) Піднесіть задане число до певного степеня.
20) Обчисліть суму елементів масиву.
21) Знайдіть максимальний елемент в заданому масиві.
22) Знайдіть найменший елемент в заданому масиві.

Лабораторна робота №4

Алгоритми сортування

І. Постановка задачі

У відповідності зі своїм варіантом необхідно реалізувати алгоритм сортування і провести експериментальне дослідження його ефективності. Розподіл алгоритмів за варіантами наведено нижче

1. Сортування обміном
2. Сортування вибором
3. Сортування включенням
4. Сортування бінарним включенням
5. Сортування Шелла
6. Сортування гребінцем
7. Сортування гнома
8. Коктейльне сортування
9. Сортування злиттям
10. Тімсорт
11. Швидке сортування
12. Сортування підрахунком
13. Блочне сортування
14. Сортування за розрядами
15. Маріонеткове сортування
16. Пірамідальне сортування
17. Інтроспективне сортування

У **звіті** повинні бути висвітлені наступні питання:

- **Опис алгоритму** – загальна характеристика (навести псевдокод або код з коментарями), його властивості (on-line, in place, stable), обчислювальна складність та складність за пам’яттю.

- **Результати експериментів** – таблиці, графіки та висновки про ефективність алгоритму.

II. Експериментальне дослідження

1) *Протестуйте алгоритм сортування на масивах різної довжини. Необхідно виміряти час роботи кожного алгоритму при різній кількості елементів в масиві — заповнити таблицю (вивести її у файл)*

Кількість елементів в масиві	Час виконання алгоритму
50 000	
100 000	
…	
1 000 000	

2) *При заповненій таблиці побудуйте для кожного алгоритму графік залежності часу його виконання від кількості елементів в масиві.*
3) *Протестуйте алгоритм сортування на масивах різної упорядкованості. Розглянути:*
    a. *найгірший та найкращий варіанти;*
    b. *масиви, заповнені псевдовипадковими числами з рівномірним розподілом з інтервалу [0, 100000];*
    c. *майже упорядковані масиви;*
    d. *масиви з певною кількістю повторень елементів;*
    e. *масиви з випадковою кількістю елементів N.*

III. Графічна реалізація

A. Елементи масиву представляються у вигляді вертикальної **гістограми**. Кожний стовпчик гістограми має висоту відповідну до значень елементів масиву. Зобразити графічно сортування масиву з *N* елементів. Стовпчики

потрібно міняти місцями наглядно (за заданим способом) на кожному кроці сортування.

B. Елементи масиву задані у вигляді **точок** на уявному графіку. Вісь **OX** представляє собою розташування елементів масиву відповідно до індексів, а вісь **OY** визначає значення самих елементів масиву. Намалювати точки та графічно відобразити їх сортування, переміщаючи їх відповідним заданим способом.

C. Елементи масиву представляються у вигляді **послідовності** невпорядкованих чисел. Зобразити графічно їх зміну місцями згідно заданому алгоритму сортування. Виділяти на кожному кроці червоним та синім кольорами кожні два елементи, що міняються місцями.

IV. Створення анімованого зображення

За серією кадрів роботи алгоритму утворити анімоване зображення.

V. Контрольні запитання

- Що таке обчислювальна складність алгоритму?
- Який алгоритм сортування називають стійким?
- Який алгоритм сортування називають сортуванням на місці?
- Яка обчислювальна складність в найгіршому випадку у алгоритму, що Ви реалізували?
- Який алгоритм вибрати для сортування масиву стрічок?
- Поясніть поведінку кривих на графіках, що Ви побудували. Чи відповідають експериментальні результати теоретичній оцінці обчислювальної складності алгоритму?
- Які алгоритми сортування не засновані на порівняннях?
- Які Ви знаєте непрактичні алгоритми сортування?

Лабораторна робота №5

Бінарні дерева та бінарні дерева пошуку

Дерева у всіх завданнях уміти виводити лівими та правими обходами в прямому, симетричному та оберненому порядках.

Бінарні дерева

1) Розробити програму, яка створює бінарне дерево T, елементами якого є дійсні числа, і обчислює добуток елементів цього дерева.
2) Розробити програму, яка створює бінарне дерево T, елементами якого є дійсні числа і знаходить найбільший та найменший елементи дерева T.
3) Розробити програму, яка створює бінарне дерево $T1$, елементами якого є дійсні числа, будує і виводить його копію $T2$.
4) Розробити програму, яка створює бінарне дерево T, елементами якого є цілі числа, знаходить і друкує всі його від'ємні елементи по рівнях.
5) Розробити програму, яка створює бінарне дерево T, елементами якого є цілі числа і знаходить довжину (кількість віток) шляху від кореня до найближчої вершини з елементом E.
6) Розробити програму, яка створює бінарне дерево T, елементами якого є дійсні числа. Підраховує середнє арифметичне всіх елементів дерева T і виводить це значення та значення його елементів.
7) Розробити програму, яка створює бінарне дерево T, елементами якого є дійсні числа, і визначає його максимальну глибину, тобто кількість віток у найдовшому шляху від кореня до листків.

Бінарні дерева пошуку

8) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка підраховує суму всіх вузлів які знаходяться на парних рівнях.
9) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка друкує всі значення дерева обходячи його по рівнях, причому по непарних рівнях рух відбувається зліва направо, а по парних навпаки.
10) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка знаходить довжину найдовшої гілки дерева.
11) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка друкує всі листові елементи дерева.
12) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка зливає два дерева в одне.
13) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка дає відповідь питання чи дерево є збалансованим.
14) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка друкє елементи найдовшої гілки деорева починаючи від листового елемента і закінчуючи коренем дерева, якщо таких гілок декілька, то надрукувати будь-яку з них.
15) Для бінарного дерева пошуку, ключами і значеннями вузлів якого є цілі числа, створити функцію яка друкє елементи найдовшої гілки деорева починаючи від кореня дерева і закінчуючи листовим елементом, якщо таких гілок декілька, то надрукувати будь-яку з них.

Лабораторна робота №6

Хеш-таблиці

Розробіть алгоритм для реалізації поставленої задачі

1. Задана така інформація про студентів: прізвище, номер групи, середній бал. Написати програму яка надає такі можливості: створення списків студентів, вставка нового студента в список, друк інформації про студента за прізвищем, збереження списку в файл, завантаження списку з файлу. Створити набір, що містить не менше 20 елементів заданих структур.
2. Створити набір, що містить не менше 20 структур інформація про слово української мови (написання кирилицею, написання латиницею). Написати програму яка буде здійснювати транслітерацію заданого тексту.
3. Створити набір, що містить не менше 20 структур інформація про місто (назва міста, список назв інших міст в які можна здійснити авіаперелiт та вартість авіаперельотів). За заданими назвами двох міст, знайти найдешевший шлях з першого міста в друге.
4. Створити набір, що містить не менше 20 структур інформація про дороги (номер дороги, назва міста в якому починається дорога, назва міста в якому закінчується дорога). Перевірити чи утворює задана послідовність номерів доріг неперервний шлях з міста в місто, якщо так, то вивести назви міст, в яких починається і закінчується цей шлях.
5. Створити набір, що містить не менше 20 структур інформація про переклад слова (мова з якої здійснюється переклад, мова на яку здійснюється переклад, слово мовою оригіналу, переклад слова). За заданим словом, мовою, до якої відноситься це слово і мовою, на яку

треба перекласти це слово, вивести його переклад. У випадку якщо це зробити неможливо – вивести повідомлення.

6. Створити набір, що містить не менше 20 структур інформація про вершину графа (ім'я вершини, список імен вершин, з якими вона з'єднується ребрами). Створити функцію CheckCycles яка повертає 1 якщо в графі є цикли і 0 у протилежному випадку.
7. Створити набір, що містить не менше 20 структур інформація про клієнтів банку (прізвище, сума грошей на рахунку). В наборі передбачити клієнтів з однаковими прізвищами, у такому випадку сума грошей додається до рахунку, або віднімається у випадку від'ємного числа. За заданим прізвищем клієнта вивести інформацію про залишок на його рахунку, якщо такого клієнта не існує – вивести відповідне повідомлення.
8. Створити хеш-таблицю «Словник синонімів», що містить не менше 20 елементів з інформацією про синоніми (слово, синонім). Здійснити заміну кожного слова його синонімів в заданому тексті. Слова в тексті, що не містяться в словнику залишити без змін. Врахувати, що в кожній парі слова є синонімами одне одного.
9. Створити хеш-таблицю «Автопарк», що містить не менше 20 елементів з інформацією про автомобілі (номер машини, марка машини, прізвище власника), використавши хеш-функцію ділення з остачею. Передбачити видалення з хеш-таблиці автомобілів, які залишили автопарк, вставлення в хеш-таблицю машин, які повернулись в автопарк, виведення хеш-таблиці.
10. Створити хеш-таблицю «Турфірма», що містить не менше 20 елементів з інформацією про подорожі (назва країни, список країн, що межують з даною країною). За введеною назвою країни необхідно запропонувати користувачу варіанти подорожі з урахуванням перетину не більше двох границь.

11. Створити хеш-таблицю «Многочлен», що містить не менше 20 елементів з інформацією про коефіцієнти многочлена (коефіцієнт, показник степеня при цьому коефіцієнті). Створити функції пошуку похідної многочлена, пошуку значення многочлена для заданого **x**, друку многочлена на екран.
12. Створити хеш-таблицю «Многочлен», що містить не менше 20 елементів з інформацією про коефіцієнти многочлена (коефіцієнт, показник степеня при цьому коефіцієнті). Створити функції для пошуку добутку та різниці двох многочленів.
13. Створити хеш-таблицю «Множина», що містить не менше 20 елементів з інформацією про множини (ім'я множини, список елементів множини). Написати програму, яка надає такі можливості: створення набору множин, добавлення множини в набір (або зміна існуючої множини), друк набору множин на екран, вилучення множини з заданим ім'ям, збереження набору множин в файл та завантаження з файлу.

Використайте одну із хеш-функцій

a) Хеш-функції KP, Add.
b) Хеш-функції KP, XOR.
c) Хеш-функції KP, FNV.
d) Хеш-функції KP, Jenkins.
e) Хеш-функції KP, ELF.
f) Хеш-функції KP, DJB (Bernstein hash).
g) Хеш-функції KP, Rotating hash.
h) Хеш-функції KP, Modified Bernstein.
i) Хеш-функції KP, Shift-Add-XOR hash.
j) Хеш-функції KP, One-at-a-Time hash.

http://www.eternallyconfuzzled.com/tuts/algorithms/jsw_tut_hashing.aspx

http://en.wikipedia.org/wiki/Jenkins_hash_function

KP – з книжки Керніган, Пайк «Практика програмування»

Завдання для літньої обчислювальної практики №1

Експериментальне дослідження ефективності реалізацій асоціативного масиву за допомогою бінарного дерева пошуку, хеш-таблиці з ланцюжками та хеш-таблиці з відкритою адресацією

Проілюструвати різні способи реалізації інтерфейсу асоціативного масиву *«Частотний словник»*. Для цього написати мовою C++ три бібліотеки для роботи з такими структурами даних:

- бінарним деревом пошуку (**Binary Search Tree),**
- хеш-таблицею з ланцюжками (**Chaining Hash Table**),
- хеш-таблицею з відкритою адресацією (**Open Addressing Hash Table**).

Ключем в кожному випадку є стрічка (**char[]**) – слово з текстового файлу, що містить не менше **200 000** слів, а *значенням* є ціле число (**int**) – лічильник кількості повторень слова. При вставці слова в структуру, якщо таке слово вже присутнє, лічильник збільшується, інакше – створюється новий елемент.

Необхідно виміряти *параметри дослідження* при кількості елементів в словнику **N=10000+i*10000**, **i=0…19**, результати у вигляді таблиці записати у файл. За інформацією з файлу побудувати графіки залежності *параметрів дослідження* від кількості елементів **N** в словнику для заданих структур даних.

	Розподіл параметрів дослідження згідно варіантів	
	Експеримент 1	**Експеримент 2**
1	Час виконання в середньому випадку функції пошуку в дереві та в хеш-таблиці з ланцюжками (використовувати двозв’язний список і хеш-функцію ELF)	Кількість колізій в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб) із хеш-функціями ADD і XOR
2	Час виконання в середньому випадку функції пошуку в дереві та в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб та хеш-функцію ADD)	Кількість колізій в хеш-таблиці з ланцюжками (використовувати однозв’язний список) із хеш-функціями FNV і KP
3	Час виконання в найгіршому випадку функції пошуку в дереві та в хеш-таблиці з ланцюжками (використовувати однозв’язний список і хеш-функцію ADD)	Час виконання функції вставки в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб) із хеш-функціями ELF і DJB
4	Час виконання в найгіршому випадку функції пошуку в дереві та в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб і хеш функцію ADD)	Час виконання функції вставки в хеш-таблиці з ланцюками (використовувати двозв’язний список) із хеш-функціями XOR і FNV

5	Час виконання функції вставки елемента в дерево та в хеш-таблицю з ланцюжками (використовувати двозв’язний список, і хеш-функцію FNV)	Час виконання в середньому випадку функції пошуку в хеш-таблиці з відкритою адресацією (метод подвійного хешування (хеш-функція XOR)) із хеш-функціями ADD і KP
6	Час виконання функції вставки елемента в дерево і в хеш-таблицю з відкритою адресацією (використовувати метод квадратичних проб, і хеш функцію FNV)	Час виконання в середньому випадку функції пошуку в хеш-таблиці з ланцюками (використовувати однозв’язний список) із хеш-функціями ELF і ADD
7	Час виконання в середньому випадку функції пошуку в хеш-таблиці з ланцюжками (використовувати однозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб) із хеш-функцією DJB	Час виконання функції пошуку мінімуму в дереві в найгіршому і середньому випадках
8	Час виконання функції вставки елемента в хеш-таблиці з ланцюжками (використовувати двозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб) і хеш функцію FNV	Час виконання функції пошуку максимуму в дереві в найгіршому і середньому випадках

9	Час виконання функції видалення в хеш-таблиці з ланцюжками (використовувати однозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб) із хеш-функцією ELF	Час виконання в середньому випадку функції пошуку в дереві
10	Час виконання в середньому випадку функції пошуку в хеш-таблиці з відкритою адресацією із хеш-функцією ELF, використовуючи метод лінійних проб та метод квадратичних проб	Час виконання функції вставки елемента в дерево
11	Кількість колізій в хеш-таблиці з відкритою адресацією і хеш функцію ADD, використовуючи метод лінійних проб та метод подвійного хешування	Час виконання функції пошуку в найгіршому випадку в дереві
12	Час виконання в середньому випадку функції пошуку в дереві та в хеш-таблиці з ланцюжками (використовувати однозв’язний список, і хеш функцію XOR)	Час виконання функції вставки в хеш-таблиці з відкритою адресацією (використовувати метод подвійного хешування (хеш-функція ADD)) із хеш-функціями FNV і KP

13	Час виконання в середньому випадку функції пошуку в дереві і в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб і хеш-функцію XOR)	Час виконання функції вставки в хеш-таблиці з ланцюками (використовувати однозв’язний список) із хеш-функціями ELF і ADD
14	Час виконання функції вставки елемента в дерево і в хеш-таблицю з ланцюжками (використовувати однозв’язний список і хеш-функцію KP)	Кількість колізій в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб) із хеш-функціями FNV і DJB
15	Час виконання функції вставки елемента в дерево і в хеш-таблицю з відкритою адресацією (використовувати метод подвійного хешування (хеш-функція FNV) і хеш функцію KP)	Кількість колізій в хеш-таблиці з ланцюками (використовувати двозв’язний список) із хеш-функціями ADD і XOR
16	Час виконання в середньому функції пошуку в хеш-таблиці з ланцюжками (використовувати двозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод лінійних проб) із хеш-функцією KP	Час виконання функції вставки в дерево

17	Час виконання функції вставки в хеш-таблиці з ланцюжками (використовувати однозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод подвійного хешування (хеш-функція ELF)) і хеш функцію KP	Час виконання в найгіршому випадку функції пошуку в дереві
18	Час виконання функції видалення в хеш-таблиці з ланцюжками (використовувати двозв’язний список) і в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб) із хеш-функцією DJB	Час виконання в найгіршому і в середньому випадках функції пошуку мінімуму в дереві
19	Час виконання в середньому функції пошуку в хеш-таблиці з відкритою адресацією із хеш-функцією XOR, використовуючи метод квадратичних проб і метод подвійного хешування (хеш-функція KP)	Час виконання в найгіршому і в середньому випадках функції пошуку максимуму в дереві

20	Кількість колізій в хеш-таблиці з відкритою адресацією із хеш-функцією DJB, використовуючи метод лінійних проб і метод квадратичних проб	Час виконання в середньому випадку функції пошуку в дереві
21	Час виконання в найгіршому випадку функції пошуку в дереві і в хеш-таблиці з ланцюжками (використовувати двозв'язний список і хеш-функцію XOR)	Час виконання в середньому випадку функції пошуку в хеш-таблиці з відкритою адресацією (використовувати метод квадратичних проб) із хеш-функціями FNV і ELF
22	Час виконання в найгіршому випадку функції пошуку в дереві і в хеш-таблиці з відкритою адресацією (використовувати метод подвійного хешування (хеш-функція ELF) і хеш функцію DJB)	Час виконання в середньому випадку функції пошуку в хеш-таблиці з ланцюками (використовувати двозв'язний список) із хеш-функціями XOR і KP

При вимірюванні часу виконання функції пошуку в середньому випадку (**Average Case**), в якості шуканого ключа вибрати випадкове слово, що вже було вставлене в частотний словник.

При вимірюванні часу виконання функції пошуку в найгіршому випадку (**Worst Case**), структуру заповнювати словами у відсортованому порядку, і в якості шуканого ключа вибрати слово, вставлене останнім.

Завдання для літньої обчислювальної практики №2

Робота з абстрактними классами

Реалізувати абстрактний базовий клас з вказаними чисто віртуальними (абстрактними) функціями і чисто віртуальними функціями вводу/виводу. Утворити похідні класи базового класу, в яких здійснити реалізацію всіх чисто віртуальних функцій. Самостійно визначити, які поля необхідні і які з них визначити в базовому класі, а які – в похідних.

1)

a) Створити базовий клас - працівник з функцію нарахування зарплати і похідні класи - службовець з погодинною оплатою, службовець в штаті і службовець з відсотковою ставкою з власною реалізацією вищенаведеної функції.

b) Створити клас **Group** (група), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку; вибірки за прізвищем та підрахунку сумарного окладу всіх працівників групи. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

2)

a) Створити абстрактний базовий клас **Norm** з віртуальними функціями обчислення норми вектора. Визначити похідні класи Vector2D (вектор на площині) і **Vector3D** (вектор у просторі) з власною реалізацією вищенаведених функцій. Для **Vector2D** модуль обчислювати як корінь з суми квадратів, для **Vector3D** як максимальне з компонентів вектора. Норму для похідних класів обчислювати як квадрат модуля.

b) Створити клас **NORMS**, що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість

виводу всіх об’єктів списку; визначення максимальної та мінімальної норм зі списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

3)

a) Створити абстрактний базовий клас **Figure** з віртуальними методами обчислення площі і периметру. Створити похідні класи: **Rectangle** (прямокутник), **Circle** (коло), **Trapezium** (трапеція) зі своїми функціями для обчислення площі і периметра.

b) Створити клас **Picture** (зображення), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку; підрахунку загальної площі та загального переметру всіх фігур групи. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

4)

a) Створити абстрактний базовий клас **Currency** (валюта) для роботи з грошовими сумами. Визначити віртуальні функції переводу суми в гривні і виводу на екран. Створити похідні класи **Dollar** (доллар) і **Euro** (євро) зі своїми функціями переводу в гривні і друку.

b) Створити клас **Purse** (гаманець), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку; виводу загальної суми, переведеної в гривні і загальної суми в кожній із валют. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

5)

a) Створити абстрактний базовий клас **Triad** з віртуальними функціями збільшення на 1 і друку значення. Створити похідні

класи **Date** (поточна дата), **Time** (поточний час) з власною реалізацією вищенаведених функцій.

b) Створити клас **Memories** (набір), що містить масив/параметризовану коллекцію пар (дата-час) об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку і визначення першої та останньої подій. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

6)

a) Створити абстрактний клас **Worker** з полями, які задають прізвище працівника, прізвища керівника і підлеглих і віртуальними методами виведення списку обов'язків та списку підлеглих на екран. На його основі реалізувати класи **Manager** (керівник проекту), **Developer** (розробник) і **Coder** (молодший програміст).

b) Створити клас **Group** (група), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку; вибірки за прізвищем з виводом всього дерева підлеглих. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

7)

a) Створити абстрактний базовий клас **Root** (корінь) з віртуальними функціями обчислення коренів і друку результату. Визначити похідні класи **Linear** (лінійне рівняння) і **Square** (квадратне рівняння) з власними функціями обчислення коренів і друку.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку. Додаткове завдання:

доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

8)

a) Створити абстрактний базовий клас **Function** (функція) з віртуальними методами обчислення значення функції в заданій точці x і виводу результату на екран. Визначити похідні класи **Ellipse** (еліпс) і **Hyperbola** (гіпербола) з власними функціями обчислення значення і друку.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

9)

a) Створити абстрактний базовий клас **Triangle** для представлення трикутника з віртуальними функціями обчислення площі і периметру. В якості параметрів класу використати дві сторони трикутника і кут між ними. Створити похідні класи прямокутний трикутник, рівнобедрений трикутник і рівносторонній трикутник зі своїми функціями обчислення площі і периметру.

b) Створити клас **Picture** (зображення), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку; підрахунку загальної площі та загального переметру всіх фігур групи. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

10)

a) Створити абстрактний клас Sorting (сортування) з ідентифікатором пос лідовності, віртуальними методами сортування, одержання суми та

виведення на екран. На його основі реалізувати класи **Choice** (метод вибору) і **Quick** (швидке сортування).

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу ідентифікаторів і сум елементів кожного об'єкту списку, а також виведення загальної суми значень. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

11)

a) Створити базовий клас список з віртуальними функціями добавлення та видалення із списку (**push**, **pop**) . Реалізувати на базі списку стек і чергу з власною реалізацією вищенаведених функцій.

b) Створити клас Series (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

12)

a) Створити абстрактний базовий клас **Integer** (ціле) з віртуальними арифметичними операціями +=, -= , *= і функцією виводу на екран. Створити похідні класи **Heximal** (шіснадцяткове число) **Binary** (двійкове число) і перевантажити для них вищевказані функції. Число має бути представлене масивом своїх цифр.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

13)

a) Створити абстрактний базовий клас **Body** (тіло) з віртуальними функціями обчислення площі поверхні і об’єму. Створити похідні класи **Parallelepiped** (паралелепіпед) і **Ball** (куля) зі своїми функціями площі поверхні і об’єму.

b) Створити клас **Picture** (зображеня), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку, підрахунку загального об’єму та площі поверхні всіх фігур. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

14)

a) Створити абстрактний клас **Point** (точка). На його основі створити класи **ColoredPoint** і **Line**. На основі класу **Line** створити клас **ColoredLine** і клас **PolyLine** (багатокутник). Всі класи повинні мати віртуальні методи установки і отримання значень всіх координат, а також зміни кольору та отримання поточного кольору.

b) Створити клас **Picture** (зображення), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

15)

a) Створити абстрактний базовий клас **Progression** (прогресія) з віртуальними функціями обчислення вказаного члена прогресії та її суми. Визначити похідні класи **Linear** (арифметична прогресія) та **Exponential** (геометрична прогресія) з власною реалізацією вищенаведених функцій.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об’єктів цих класів в динамічній пам’яті. Передбачити можливість виводу всіх об’єктів списку та обчислення загальної суми

всіх прогресій. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

16)

a) Створити абстрактний базовий клас **Pair** з віртуальними арифметичними операціями +=, -=, *=. Створити похідні класи Complex (комплексне число) і Rational (раціональний дріб) з власною реалізацією арифметичних операцій.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

17)

a) Описати абстрактний клас **Element** (елемент логічної схеми), задавши в ньому числовий ідентифікатор, кількість входів, ідентифікатори приєднаних до нього елементів (до 10) і виконавчі значення на входах і виході. На його основі реалізувати класи **AND** і **OR** , які можуть мати різну кількість входів і один вихід і реалізують логічне множення додавання відповідно.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

18)

a) Створити абстрактний базовий клас **Array** з віртуальними методами знаходження суми елементів і поелементній обробці елементів масиву foreach(). Створити похідні класи **SortArray** і **XorArray**. Для **SortArray** знаходження суми елементів реалізується як додавання множин, а **foreach** – сортування. Для **XorArray** знаходження суми

елементів реалізується як побітове виключаючи **АБО** для елементів масивів, а **foreach** – нормування елементів масиву.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

19)

a) Створити абстрактний базовий клас базовий клас **Array** з віртуальними методами знаходження суми елементів і поелементній обробці елементів масиву **foreach**(). Створити похідні класи **AndArray** і **OrArray**. Для **AndArray** знаходження суми елементів реалізується як перетин множин, а **foreach** – нормування елементів масиву. Для **OrArray** знаходження суми елементів реалізується як об'єднання множин, а **foreach** – реалізовує швидке сортування.

b) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

20)

a) Створити абстрактний базовий клас **Container** з віртуальними методами **sort()** і обробці елементів контейнера **norma()**. Створити похідні класи **Bubble** і **Choice**. Для **Bubble** метод **sort** має реалізовувати бульбашкове сортування, а norm обчислювати квадратний корінь з суми елементів. Для Choice метод sort має реалізовувати сортування методом вибору максимального елемента, а norma середнє арифметичне всіх елементів.

a) Створити клас **Series** (набір), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку. Додаткове завдання:

доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

21)

a) Створити абстрактний базовий клас **Body** (тіло) з віртуальними функціями обчислення площі поверхні і об'єму. Створити похідні класи **Cube** (куб) і **Piramide** (піраміда) зі своїми функціями площі поверхні і об'єму.

b) Створити клас **Picture** (зображеня), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку, підрахунку загального об'єму та площі поверхні всіх фігур. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

22)

a) Створити абстрактний базовий клас **Vehicle**. Створити похідні класи **Car** (автомобиль), **Bicycle** (велосипед) и **Lorry** (грузовик). Класи повинні мати можливість задавати і отримувати параметри засобів пересування (ціна, максимальна швидкість, рік випуску і т.д.). Поряд із загальними полями та методами, кожен клас має мати і специфічні для нього поля та методи.

b) Створити клас **Garage** (гараж), що містить масив/параметризовану коллекцію об'єктів цих класів в динамічній пам'яті. Передбачити можливість виводу всіх об'єктів списку, підрахунку загального об'єму та площі поверхні всіх фігур. Додаткове завдання: доповини клас методами сортування за деяким критерієм, виводу у файл та зчитування з файлу.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. **Ахо А.** *Структуры данных и алгоритмы* / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М.: Изд. Дом «Вильямс», 2001. — 384с.
2. **Браунси К.** *Основные концепции структур данных и реализация в C++* / К. Браунси. – М.: Изд. Дом «Вильямс», 2002. – 320с.
3. **Вирт Н.** *Алгоритмы и структуры данных* / Н. Вирт. — М., ДМК_Пресс, 2011. — 272 с.
4. **Кнут Д.** *Искусство программирования, т.3. Сортировка и поиск* / Д. Кнут. — М.: Вильямс, 2011. — 824 с.
5. **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.
6. **Романовский И. В.** *Вычислительная математика и структура алгоритмов* / И.В. Романовский. — М.: МГУ, 2006. — 112 с.
7. **Страуструп Б.** *Язык программирования C++. Специальное издание* / Б. Страуструп: Пер. с англ. — М.: Издательство Бином, 2011 г. — 1136 с.

Навчальне видання

Інесса Краснокутська, кандидат фізико-математичних наук

АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ

Лабораторний практикум

для студентів спеціальності
8.04030101 «Прикладна математика»
денної та заочної форм навчання

Відповідальний за випуск завідувач кафедри прикладної математики
та інформаційних технологій професор Я.Й. Бігун
Комп'ютерне верстання, літературне та технічне редагування
І.В. Краснокутської та О.С. Краснокутського

Підписано до друку 01.09.2017. Формат 60x84/16.
Папір офсетний. Друк різографічний. Ум. друк. арк. 2.
Обл. вид. арк. 2. Тираж 25.
Видавництво та друкарня Чернівецького національного університету.
58012, Чернівці, вул. Коцюбинського, 2.
Свідоцтво суб'єкта видавничої справи ДК № 891 від 08.04.2002.