МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт студентів з дисципліни
“ ***Алгоритмізація та програмування*** ”
Частина 2
для студентів II року денної форми навчання
Спеціальність – 122 «Комп’ютерні науки»

Одеса 2017

Методичні вказівки до виконання лабораторних робіт студентів з дисципліни “Алгоритмізація та програмування ”, частина 2, для студентів І року денної форми навчання. Спеціальність – 122 «Комп’ютерні науки» / Укладачі: Кузніченко С.Д., к.г.н, доц., Коваленко Л.Б., доц. – Одеса, ОДЕКУ, 2017. - 50 с.

Зміст

Передмова

Методичні вказівки призначені для студентів І курсу денної форми навчання. Мета виконання лабораторних робіт – закріплення теоретичного лекційного матеріалу та придбання практичних навичок програмування мовою Java, яка є сучасною та найбільш розвиненою у цей час.

Дисципліна «Алгоритмізація та програмування» є обов'язковою дисципліною на рівні вищої освіти бакалавр за спеціальністю «Комп'ютерні науки». Внаслідок вивчення даної дисципліни студенти повинні **знати** найпростіші поняття ООП: класи, об'єкти, властивості (поля) і дії (методи), структури даних і алгоритми, основи роботи з рядками, ведення-виведення у/з файл(у), динамічні структури даних (списки, черги, стеки і бінарні дерева) та алгоритми їх оброблення. Вони повинні **вміти** застосовувати алгоритми подання, зберігання і обробки інформації, складати лінійні та розгалужені програми мовою Java та здобути практичні навички створення і модифікації невеликих програмних проектів.

Методичні вказівки містять рекомендації по вивченню розділів дисципліни, контрольні запитання та завдання. Всі лабораторні роботи підкріплені прикладами розв'язання типових задач на ПЕОМ.

Під час підготовки до лабораторної роботи студент повинен вивчити відповідний теоретичний матеріал за конспектом лекцій і літературою, розібрати приклади розв'язання задач, та відповісти на контрольні питання. Виконанню лабораторної роботи передує практичне заняття з відповідної теми. На практичних заняттях розглядаються алгоритми рішення задач з тим, щоб під час лабораторної роботи студенти складали програму, користуючись розглянутими алгоритмами.

На початку лабораторної роботи викладач проводить співбесіду за результатами якої студент отримує, або не отримує допуск до виконання лабораторної роботи. Якщо студент не отримав допуску, він залишається на заняттях, але не виконує лабораторної роботи на комп'ютері. Замість цього він вивчає теоретичний матеріал за даною темою, щоб відповісти на питання викладача та отримати допуск до виконання роботи.

За кожну лабораторну роботу студент отримує дві оцінки: за виконання та за захист роботи. Згідно з цих пунктів студенту зараховується відповідна кількість балів. Максимальні бали з кожної лабораторної роботи встановлюються згідно робочої програми дисципліни. На першому занятті студенти отримують графік контролюючих заходів: перелік контролюючих заходів, терміни виконання, бали за кожний вид робіт.

Список літератури

Основна

1. *Верлань А.Ф., Чмырь И.А., Кузніченко С.Д., Коваленко Л.Б.* Императивное программирование и объектно-ориентированное моделирование: Java, UML, OCL : Учебное пособие для студентов высших учебных заведений. Одесса, Издательство Экология. – 2014., 326 с.
2. *Кузніченко С.Д., Коваленко Л.Б.* Алгоритмізація та програмування: Конспект лекцій – Одеса: (ел.вар), 2015. – 340 с.

Додаткова

3. *Эккель Б.* Философия Java.: BHV., СПб – 2001. 850с.
4. *Глушаков С.В*. Программирование на Java 2: Изд.2-е.- Харьков: Фолио, 2003. – 536 с. – (Учебный курс).
5. *А.В. Картузов, Д.В. Николенко*. Программируем на языке Java – СПб: Наука и техника, 2001. – 192 стр. с ил.
6. *Любош Бруга.* Java по-быстрому. Практический экспресс-курс – СПб: Наука и техника, 2006. – 384 с. :ил.
7. *Аккуратов Е.Е.* Знакомьтесь: Java. Самоучитель. – М.: Изд. Дом «Вильямс», 2006. – 256с.: ил.

Лабораторна робота № 1

Тема: «Робота з двовимірними масивами»

1. Мета роботи

Метою лабораторної роботи є отримання практичних навичок програмування алгоритмів обробки двовимірних масивів. Вивчення алгоритмів упорядкування двовимірних масивів методом вставки, вибору та лінійного сортування.

2. Завдання до лабораторної роботи

Використовуючи алгоритми, розглянуті на практичному занятті, скласти програму розрахунку заданих величин.

2.1 Методичні вказівки

■ Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних тем лекційного курсу:

- Багатомірні масиви.
- Приклади рішення завдань із використанням багатомірних масивів.

Тому під час підготовки до лабораторної роботи рекомендується повторити зазначені розділи дисципліни.

■ Розглянемо фрагменти програм, де використовуються алгоритми сортувань двовимірних масивів за рядками і за стовбцями:

- ***сортування методом вибору рядків матриці за зростанням максимальних елементів у рядках***

```
/* Оголошення двовимірного масиву*/
double a[][]=new double[n][m];
. . .
/* Знаходження максимальних елементів за рядками*/
double max[]=new double[a.length];
. . .
// Сортування матриці методом вибору
double c[]=new double[m];// допоміжний масив, для
                         // проміжного зберігання
int nom;
double min;
 for( int i=0; i < a.length-1; i++) {
```

```
        nom=i; min=max[i];
      for(int j=i+1; j < a.length; j++)
      if (  max[j] < min ) {
       nom=j; min=max[j]; }
        max[nom] = max[i]; max[i] = min;
         c=a[nom]; a[nom]=a[i]; a[i]=c;}
    . . .
```

- сортування методом вставки рядків матриці за убуванням максимальних елементів у рядках

```
    . . .
double x; int j;
  for ( int i=1; i < n; i++) {
    x = max[i]; c=a[i];
    for ( j=i-1; j>=0 && max[j] < x; j--)
      {max[j+1] = max[j]; a[j+1]=a[j];}
      max[j+1] = x; a[j+1]=c; }
    . . .
```

- сортування методом лінійного сортування рядків матриці за зростанням максимальних елементів у рядках

```
    . . .
    int i, j;
      double x;
      for( i=0; i < n-1; i++) {
        for( j = i+1; j < n; j++) {
          if ( max[i] > max[j] ) {
       x=max[i]; max[i]=max[j]; max[j]=x;
          c=a[i]; a[i]=a[j]; a[j]=c;
        }}}
    . . .
```

- сортування методом вибору стовпців матриці за зростанням максимальних елементів у стовпцях

```
    /* Оголошення двовимірного масиву*/
    double a[][]=new double[n][m];
    . . .
    /* Знаходження максимальних елементів за стовпцями*/
    double max[]=new double[m];
    for(int i=0;i<m;i++){
    max[i]=a[0][i];
    for(int j=0;j<n;j++){
    if (a[j][i]>max[i]) max[i]=a[j][i];}}

    /* Сортування масиву методом вибору*/
    double c[]=new double[n];
    int nom;
    double min;
```

```
      for( int i=0; i < m-1; i++) {
        nom=i; min=max[i];
        for(int j=i+1; j < m; j++)
          if (  max[j] < min ) {
            nom=j; min=max[j];
          }
        max[nom] = max[i]; max[i] = min;
        for(int k=0;k<n;k++)
        {c[k]=a[k][nom]; a[k][nom]=a[k][i]; a[k][i]=c[k];}}
    . . .
```

- сортування методом вставки стовпців матриці за убуванням максимальних елементів у стовпцях

```
    . . .
    double x; int j;
     for ( int i=1; i < m; i++) {
      x = max[i];
      for ( int k=0; k < n; k++)
       c[k]=a[k][i];
      for ( j=i-1; j>=0 && max[j] < x; j--)
          {max[j+1] = max[j];
       for ( int k=0; k < n; k++)
              a[k][j+1]=a[k][j];}
        max[j+1] = x;
        for ( int k=0; k < n; k++)
        a[k][j+1]=c[k];}
    . . .
```

- сортування методом лінійного сортування стовпців матриці за зростанням максимальних елементів стовпцях

```
    . . .
int i, j;
  double x;
  for( i=0; i < m-1; i++) {
    for( j = i+1; j < m; j++) {
      if ( max[i] > max[j] ) {
   x=max[i]; max[i]=max[j]; max[j]=x;
      for( int k=0; k < m-1; k++)
      {c[k]=a[k][i]; a[k][i]=a[k][j]; a[k][j]=c[k];}
    }}}
    . . .
```

2.2 Приклад програми

Дано двовимірний цілий масив. Упорядкувати рядки матриці за убуванням кількості додатних елементів у рядку. Сортування методом вибору. Знайти суму парних елементів у кожному стовпці матриці.

```
import java.util.*;
class Lab1{
public static void main(String[] args){
Scanner scan = new Scanner (System.in);
System.out.println("Кількість рядків:");
int n=scan.nextInt();
System.out.println("Кількість стовпців:");
int m=scan.nextInt();
int a[][]=new int[n][m];

int count[]=new int[a.length];
int sum[]=new int[m];
int c[]=new int[m];

// Заповнення елементів матриці за допомогою генератору
//випадкових чисел
for(int i=0;i<a.length;i++)
for(int j=0;j<a[i].length;j++)
a[i][j]=(int)(Math.random()*20-10);
System.out.println("Вихідний двовимірний масив ");
for(int i=0;i<a.length;i++){
for(int j=0;j<a[i].length;j++)
System.out.printf("%8d",a[i][j]);
System.out.println();}

// Знаходження кількості додатних елементів за рядками
for(int i=0;i<a.length;i++){
count[i]=0;
for(int j=0;j<a[i].length;j++)
if (a[i][j]>0) count[i]++;}

// Сортування масиву методом вибору
int nom, min;
  for( int i=0; i < a.length-1; i++) {
    nom=i; min= count[i];
    for(int j=i+1; j < a.length; j++)
      if (  count[j] > min ) {
        nom=j; min= count[j];
      }
    count[nom] = count[i]; count[i] = min;
    c=a[nom]; a[nom]=a[i]; a[i]=c;
}
  // Знаходження суми парних елементів у кожному стовпці
 for(int i=0;i<m;i++){
     sum[i]=0;
for(int j=0;j<n;j++)
if(a[j][i]%2==0) sum[i]+=a[j][i];
}
// Виведення упорядкованого масиву
System.out.printf("%s%40s%n","Відсортований двовимірний
масив","Кількість дод. елементів");
```

```
for(int i=0;i<a.length;i++){
for(int j=0;j<a[i].length;j++){
System.out.printf("%8d",a[i][j]);}
System.out.printf("%16d%n",count[i]);}
System.out.println("Сума парних елементів за стовпцями ");
for(int j=0;j<m;j++)
System.out.printf("%8d",sum[j]);
}}
```

2.3 Контрольні питання

1. Як оголошуються в Java багатомірні масиви? Скільки одномірних масивів створюється при оголошенні `int x[][]=new int[3][5];`?
2. Як здійснюється доступ до елементів двовимірного масиву?
3. Чому буде дорівнюватися змінна len після наступного оголошення та ініціалізації масиву x

```
int x[][] = {{-1,2,3}, null, {0,3}, {8,0,4}};
int len=x[3].length/ x[2].length;
```

4. Чому буде дорівнюватися змінна len після наступного оголошення й ініціалізації масиву x

```
int x[][]=new int[3][5];
x[0]=new int[7];
x[1]=new int[2];
x[2]=null;
int len= x.length+x[1].length;
```

5. Як перетворити одномірний масив у двовимірний і навпаки?

Варіанти завдань

1. Дано двовимірний дійсний масив. Упорядкувати рядки матриці за зростанням максимальних елементів у рядках. Сортування методом вибору. Знайти добуток елементів, які за модулем більше 2, у кожному стовпці матриці.

2. Дано двовимірний дійсний масив. Упорядкувати рядки матриці за убуванням суми додатних елементів у рядках. Сортування методом вставки. Знайти мінімум у кожному стовпці матриці.

3. Дано двовимірний цілий масив. Упорядкувати рядки матриці за зростанням початкових елементів у рядках. Сортування методом лінійного сортування. Знайти максимальний за модулем елемент у

кожному рядку матриці. Знайти для всієї матриці кількість елементів, які кратні 7.

4. Дано двовимірний цілий масив. Упорядкувати рядки матриці за убуванням кількості непарних елементів у рядках. Сортування методом вибору. Знайти максимальний для всієї матриці елемент і номер рядка й стовпця, у якому він знаходиться.

5. Дано двовимірний дійсний масив. Упорядкувати рядки матриці за зростанням суми від'ємних елементів у рядку. Сортування методом лінійного сортування. Знайти максимум у кожному стовпці матриці.

6. Дано двовимірний цілий масив. Упорядкувати рядки матриці за убуванням початкових елементів у рядках. Сортування методом вставки. Знайти максимальний елемент у кожному стовпці й кількість непарних елементів у всій матриці.

7. Дано двовимірний дійсний масив. Упорядкувати рядки матриці за убуванням мінімальних елементів у рядках. Сортування методом вставки. Знайти кількість від'ємних елементів у кожному стовпці матриці.

8. Дано двовимірний цілий масив. Упорядкувати рядки матриці за зростанням кількості парних елементів у рядках. Сортування методом лінійного сортування. Знайти мінімальний елемент серед додатних елементів у кожному стовпці матриці.

9. Дано двовимірний дійсний масив. Упорядкувати рядки матриці за убуванням кількості нульових елементів у рядках. Сортування методом вибору. Знайти максимальний елемент серед від'ємних елементів у кожному стовпці матриці.

10. Дано двовимірний дійсний масив. Упорядкувати стовпці матриці за зростанням максимальних елементів у стовпцях. Сортування методом вставки. Знайти середнє арифметичне значення елементів у кожному рядку матриці.

11. Дано двовимірний цілий масив. Упорядкувати стовпці матриці за убуванням середніх арифметичних значень елементів у стовпцях. Сортування методом вибору. Знайти добуток додатних елементів у кожному рядку матриці.

12. Дано двовимірний дійсний масив. Упорядкувати стовпці матриці за зростанням кількості елементів, що лежать у діапазоні від A до B, у стовпцях. Сортування методом лінійного сортування. Знайти максимум у кожному рядку матриці.

13. Дано двовимірний дійсний масив. Упорядкувати стовпці матриці за убуванням мінімальних за модулем елементів у стовпцях. Сортування методом вибору. Знайти кількість ненульових елементів у кожному рядку матриці.

14. Дано двовимірний цілий масив. Упорядкувати стовпці матриці за зростанням добутку елементів у стовпцях. Сортування методом вставки. Знайти мінімум у кожному рядку матриці.

15. Дано двовимірний дійсний масив. Упорядкувати стовпці матриці за убуванням максимальних за модулем елементів у стовпцях. Сортування методом лінійного сортування. Знайти добуток парних елементів у кожному рядку матриці.

3. Прилади, устаткування та інструменти

Для виконання лабораторної роботи використовується ПЕОМ з установленим пакетом Sun Microsystems JDK 1.5 і вище та інтегрованим середовищем розробки BlueJ або Eclipse. Для написання програми на Java може бути використаний будь-який текстовий редактор, наприклад, Notepad, WordPad в MS Windows і ін.

4. Правила техніки безпеки та охорони праці

Правила техніки безпеки при виконанні лабораторної роботи регламентуються «Правилами техніки безпеки при роботі в комп'ютерній лабораторії».

5. Порядок проведення лабораторної роботи

Для виконання роботи кожен студент повинен:

1. Відповісти на контрольні питання та пройти усне опитування за теоретичним матеріалом лабораторної роботи;
2. Пройти інструктаж за правилами охорони праці;

3. Отримати варіант завдання у викладача;
4. Скласти алгоритм розв’язання задачі;
5. Записати код програми на комп’ютері;
6. Відкомпілювати програму та виправити всі помилки;
7. Запустити програму на виконання;
8. Отримати результати роботи програми і показати їх викладачу;
9. Підготувати і захистити звіт до лабораторної роботи.

6. Оформлення і захист звіту

Підготовлений до захисту звіт до лабораторної роботи повинен містити:

1. титульний лист, де вказані номер і назва лабораторної роботи, відомості про виконавця;
2. номер варіанта роботи та текст завдання;
3. відповіді на контрольні запитання до лабораторної роботи;
4. текст програми алгоритмічною мовою Java;
5. лістинг результатів виконання програми.

Лабораторна робота № 2

Тема: «Оголошення і виклик методів. Передача в метод параметрів примітивних типів. Рекурсивні методи»

1. Мета роботи

Отримання практичних навичок оголошення і виклику статичних методів. Вивчення особливостей створення рекурсивних методів.

2. Завдання до лабораторної роботи

1. Створіть клас, що містить метод, який дозволяє з точністю $Eps=1\cdot10^{-4}$ розрахувати таблицю значень функції дійсної змінної x. Результати роботи порівняйте з результатами розрахунку відповідного методу класу Math.
2. При рішенні другого завдання використайте **рекурсивний метод**.

2.1 Методичні вказівки

■ Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних тем лекційного курсу:

- Методи. Оголошення і виклик методів. Сигнатура методу.
- Модифікатори членів класу.
- Передача аргументів за посиланням та за значенням.
- Рекурсія. Особливості рекурсивних програм.

Тому під час підготовки до лабораторної роботи рекомендується повторити зазначені розділи дисципліни.

Наведемо нижче декілька важливих визначень, які слід пам’ятати під час виконання лабораторної роботи.

■ **Методи** - це підпрограми, які приєднані до конкретних визначень класів. Вони описуються усередині визначення класу на тому ж рівні, що і змінні класу. При оголошенні методу задаються тип результату, що повертається ним, і список параметрів. Загальна форма оголошення методу має наступний вигляд:

тип ім’я _методу (список формальних параметрів) { тіло методу; }

■ **Сигнатура** (signature) методу визначається ім'ям методу і його параметрами (кількістю, типом та порядком слідування). Якщо для полів забороняється збіг імен, то для методів у класі заборонене створення двох методів з однаковими сигнатурами. Пам’ятайте, що тип значення, що

повертає метод, не входить у сигнатуру, тому методи не можуть розрізнятися тільки типом результату їхньої роботи.

■ **Модифікатори** – службові слова, які можуть використовуватися перед описом поля або методу класу.

1. Додаванням модифікатора **private** до опису члена класу, дозволяє зробити його закритим, тобто доступ до таких членів класу (полів або методів) буде дозволений тільки усередині того класу, в якому вони оголошені.

2. У протилежність закритості можна оголосити деякі члени класу відкритими, записавши замість слова private модифікатор **public**. До таких методів і змінних дозволяється доступ з будь-якого класу.

3. Іноді треба визначити поле, загальне для всього класу, зміна якого в одному екземплярі викличе зміну того ж поля у всіх екземплярах. Такі поля називаються **змінними класу** (class variables). Для змінних класу виділяється тільки одна комірка пам'яті, загальна для всіх екземплярів. Змінні класу в Java утворюються модифікатором **static**.

Для **статичних методів** виконавча система Java завжди створює в пам'яті тільки одну копію машинного коду методу. Статичні методи виконуються відразу у всіх екземплярах класу. Більш того, вони можуть виконуватися, навіть якщо не створений жоден екземпляр класу. Досить уточнити ім'я методу ім'ям класу (а не ім'ям об'єкту), щоб метод міг працювати. Саме так ми користувалися методами класу Math, не створюючи його екземпляри, а просто записуючи Math.abs(x), Math.sqrt(x).

Тому статичні методи називаються **методами класу**, на відміну від нестатичних методів, які звуться **методами екземпляра**.

■ **Рекурсія** - це такий спосіб організації обробки даних, при якому програма викликає сама себе безпосередньо, або за допомогою інших програм. Метод P називається **рекурсивним**, якщо при виконанні тіла методу відбувається виклик методу P.

Прикладами рекурсивних функцій можуть служити факторіал числа і числа Фібоначчі. У кожному з цих випадків значення функції для всіх значень аргументу, починаючи з деякого, визначається через попередні значення. Вирішуючи деяку задачу, рекурсивний алгоритм викликає сам себе для вирішення її підзадач.

■ Розглянемо декілька прикладів розв’язання задач за допомогою рекурсивних методів:

Рекурсивне обчислення значення факторіала натурального числа

Факторіал n! цілого невід’ємного числа n задається наступними співвідношеннями: $0!=1$, $n!=n\cdot(n-1)!$ для $n>0$

```
   import java.util.*;
   public class FactN {
   static int fac(int x) { return (x == 0) ? 1 : x * fac(x-1);}

public static void main(String[] args){
Scanner scan = new Scanner(System.in);
System.out.print("n=");
System.out.printf("n!=%d",fac(scan.nextInt())); }}
```

Рекурсивне обчислення чисел Фібоначчі

Числами Фібоначчі f_n називають послідовність величин 0, 1, 1, 2, 3, 5, 8, ..., яка визначається наступним чином:

$f_0 = 0,$

$f_1 = 1,$ для n>1

$f_n = f_{n-1} + f_{n-2}$

```
import java.util.*;
public class FibN {
static int fib(int x) {
return (x > 1) ? fib(x-2) + fib(x-1) : (x == 1) ? 1 : 0; }

public static void main(String[] args){
Scanner scan = new Scanner(System.in);
System.out.print("Введіть n ->");
int n = scan.nextInt();
System.out.printf("fib(%d) = %d",n,fib(n));}}
```

Рекурсивне обчислення найбільшого спільного дільника двох чисел

```
import java.util.*;
public class Nsd {
 static int nsd(int x, int y) {
 return (x==y)? x:(x>y)?nsd(x-y,y):nsd(x,y-x); }

public static void main(String[] args){
  Scanner scan = new Scanner(System.in);
   System.out.print("Введіть a ->");
   int a = scan.nextInt();
   System.out.print("Введіть b ->");
   int b = scan.nextInt();
   System.out.printf("НСД = %d",nsd(a,b));
     }}
```

Рекурсивний метод піднесення до ступеня

```
import java.util.*;
```

```
public class Rec {
      static int step(int x, int n) {
      return (n==0)?1:x*step(x,n-1); }

public static void main(String[] args){
  Scanner scan = new Scanner(System.in);
   System.out.print("Введіть a ->");
    int a=scan.nextInt();
   System.out.print("Введіть n ->");
    int n=scan.nextInt();
   System.out.printf("Значення %d^%d=%d",a,n,step(a,n));
     }}
```

2.2 Приклад програми

∗ Розрахуйте

$$\ln x = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \cdots + (-1)^{n+1}\frac{(x-1)^n}{n},$$

де $0<x\leq 2$,

```
public class Zad1 {
static final double eps=1e-4;

       public static double my_log(double x) {
       double sum=x-1,c=x-1;
        int n=1;
        while(Math.abs(c/n)>eps){
        c*=-(x-1); sum+=c/++n; }
       return sum;
}
public static void main(String[] args) {
System.out.printf("------------------------------%n");
System.out.printf("|    x    | my_log(x)| log(x) |%n");
System.out.printf("------------------------------%n");
   for(double x=0.1;x<=2;x+=(2-0.1)/5){
   System.out.printf("|%8.4f |%8.4f |%8.4f |%n", x, my_log(x),
Math.log(x));}
System.out.printf("------------------------------%n");
    }}
```

```
------------------------------
|    x    | my_log(x)| log(x) |
------------------------------
|  0,1000 | -2,3019 | -2,3026 |
|  0,4800 | -0,7339 | -0,7340 |
|  0,8600 | -0,1508 | -0,1508 |
|  1,2400 |  0,2151 |  0,2151 |
|  1,6200 |  0,4824 |  0,4824 |
|  2,0000 |  0,6931 |  0,6931 |
------------------------------
```

∗ Обчисліть (у виразі присутні n радикалів): $\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}$

```
import java.util.*;
public class Rec {
    static double fac(int n) {
    double f;
    return (n==1)?Math.sqrt(2):Math.sqrt(2+fac(--n)); }

 public static void main(String[] args) {
  Scanner scan = new Scanner(System.in);
 System.out.print("Введіть n:");
 System.out.printf("Значення=%f",fac(scan.nextInt()));
           }}
```

Перевірка: при n=3, вираження дорівнює 1.9615705608…

2.3 Контрольні питання

1. Приведіть формат оголошення методу.
2. Що таке сигнатура методу? Які з наведених нижче методів несумісні в одному класі?
```
void get() {}
int get() {}
void get(int x) {}
void get(int y) {}
public int get() {}
private int get() {}
```
3. У яких випадках у тілі методу використовується return-вираження?
4. Чим відрізняються формальні і фактичні параметри?
5. Дайте визначення глобальним і локальним змінним. Приведіть приклади.
6. У яких випадках при оголошенні полів і методів класу використовуються наступні модифікатори доступу: private, public, static, final?

2.4 Варіанти завдань

1.

∗ $\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\cdots\right)$, при |x|>1

де $2\le x\le 5$

∗ Напишіть програму обчислення

$$\sum_{k=1}^{n}(-1)^k \cdot k!!, \text{где } k!! = \begin{cases} 1\cdot 3\cdot 5\cdot \ldots \cdot k, & \text{якщо } k - \text{непарне} \\ 2\cdot 4\cdot 6\cdot \ldots \cdot k, & \text{якщо } k - \text{парне} \end{cases}$$

Використовуйте рекурсивний метод обчислення k!!

2.

$$* \; e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots \text{, при } |x| < \infty$$

де $0.5 \le x \le 0.9$

∗ Скласти програму обчислення суми: 2! - 4! + 6! - ... + n! ($n \le 16$, n – парне). Використовуйте рекурсивний метод обчислення n!

3.

$$* \; \ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots \text{, при } -1 < x \le 1$$

де $-0.5 \le x \le 0.5$

∗ Числа Фібоначчі $u_0, u_1, u_2, \ldots$ визначаються наступним чином: $u_0=0$; $u_1=1$; $u_n=u_{n-1}+u_{n-2}$, n=2,3,4,... Напишіть програму обчислення першого числа Фібоначчі, що більше за m (m>1). Використовуйте рекурсивний метод, заснований на безпосередньому використанні співвідношення $u_n=u_{n-1}+u_{n-2}$.

4.

$$* \; \ln\frac{1+x}{1-x} = 2\sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots \right), \text{ при } |x| < 1$$

де $-0.8 \le x \le 0.8$

∗ Знайти двозначні числа (за винятком тих чисел, що закінчуються на нуль 10, 20, 30,…), у яких НСД цифр рівний 3. Використовуйте рекурсивний метод знаходження найбільшого спільного дільника двох натуральних чисел.

5.

$$* \; \ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots \right), \text{ при } -1 \le x < 1$$

де $-0.5 \le x \le 0.8$

∗ Знайдіть всі тризначні числа, які можна представити у вигляді суми факторіалів їх цифр. Використовуйте рекурсивний метод обчислення n!

6.

∗ $\text{arcctg}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + \dots$, при $|x| \le 1$

де $-1 \le x \le 0.5$

∗ Числа Фібоначчі $u_0, u_1, u_2, \dots$ визначаються наступним чином: $u_0=0$; $u_1=1$; $u_n=u_{n-1}+u_{n-2}$, $n=2,3,4,\dots$ Знайдіть суму перших n чисел Фібоначчі ($n<=10$). Використовуйте рекурсивний метод, заснований на безпосередньому використанні співвідношення $u_n=u_{n-1}+u_{n-2}$.

7.

∗ $\text{arctg}(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} \dots$, при $x>1$

де $\frac{\pi}{2} \le x \le \pi$

∗ Опишіть рекурсивний метод піднесення до ступеня pow(x,n), формальними параметрами якого є дійсна змінна x і натуральна змінна n, використовуйте його для обчислення значення виразу $\sum_{k=1}^{10} k^k$

8.

∗ $e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots$, при $|x| < \infty$

де $0.2 \le x \le 0.8$

∗ Серед перших n+1 елементів послідовності

$$\begin{cases} x_0 = a \\ x_k = q \cdot x_{k-1} + b, \text{ де } k = 1,2,\dots \end{cases}$$

знайти кількість тих елементів, які належать інтервалу (c,d).

9.

∗ $\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$, при $|x| < \infty$

де $0.1 \le x \le 0.8$

∗ Числа Фібоначчі $u_0, u_1, u_2, \dots$ визначаються наступним чином: $u_0=0$;

$u_1=1$; $u_n=u_{n-1}+u_{n-2}$, $n=2,3,4,...$ Серед перших 10-ти чисел Фібоначчі знайдіть два таких, які в сумі дорівнюють 42. Використовуйте рекурсивний метод, заснований на безпосередньому використанні співвідношення $u_n=u_{n-1}+u_{n-2}$.

10.

$$* \frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\ldots, \text{ при } |x|<\infty$$

де $\frac{\pi}{6}\le x\le\frac{\pi}{4}$

∗ Відомо, що $x_0=a$, $x_k=qx_{k-1}+b$, де $k=1,2,...$ Обчисліть x_n для заданого натурального n.

11.

$$* \ln x=2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}=2\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+\ldots\right), \; x>0$$

де $2\le x\le 5$

∗ Напишіть програму обчислення P за формулою:

$$p=\begin{cases}(n!+4)^2, & \text{якщо } n<5\\ n!, & \text{якщо } n\ge 5\end{cases}$$

де n - задане натуральне число. Використовуйте рекурсивний метод обчислення n!

12.

$$* \ln x=\sum_{n=0}^{\infty}\frac{(x-1)^{n+1}}{(n+1)(x+1)^{n+1}}=\frac{x-1}{x}+\frac{(x-1)^2}{2x^2}+\frac{(x-1)^3}{3x^3}+\ldots, \text{ при } x>\frac{1}{2}$$

де $0.6\le x\le 0.9$

∗ Числа Фібоначчі $u_0,u_1,u_2,...$ визначаються наступним чином: $u_0=0$; $u_1=1$; $u_n=u_{n-1}+u_{n-2}$, $n=2,3,4,...$ Серед перших 10-ти чисел Фібоначчі знайдіть кількість чисел, що належать відрізку $[5 \div 45]$. Використовуйте рекурсивний метод, заснований на безпосередньому використанні співвідношення $u_n=u_{n-1}+u_{n-2}$.

13.

$$* \operatorname{Arth}(x)=\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ldots, \text{ при } |x|<1$$

де $0.2\le x\le 0.5$

Зауваження: Arth(x) – зворотний гіперболічний тангенс;

$\mathrm{Arth}(x)=\frac{1}{2}\ln\frac{1+x}{1-x}$, при |x|<1

∗ Опишіть рекурсивний метод піднесення до ступеня pow(x,n), формальними параметрами якого є дійсна змінна x і натуральна змінна n, використовуйте його для обчислення суми $\sum_{i=1}^{10}\frac{x^i}{i}$.

14.[*]

$$\arcsin(x)=x+\sum_{n=1}^{\infty}\frac{1\cdot 3\cdot\ldots\cdot(2n-1)\cdot x^{2n+1}}{2\cdot 4\cdot\ldots\cdot 2n\cdot(2n+1)}=x+\frac{x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+\frac{1\cdot 3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7}+\ldots$$

при |x|<1
де 0.1≤x≤0.3

∗ Числа Фібоначчі $u_0,u_1,u_2,...$ визначаються таким чином: ($u_0=0$; $u_1=1$; $u_n=u_{n-1}+u_{n-2}$, n=2,3,4,... Перевірте на декількох прикладах, чи буде u_{5k}, де k=1,2,3,4,5 ділитися на 10. Використовуйте рекурсивний метод заснований на використанні співвідношення $u_n=u_{n-1}+u_{n-2}$.

15.

∗ $\mathrm{arctg}(x)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots$, при |x|≤1

де -1≤x≤1

∗ Скласти програму обчислення суми: 1! - 2! + 3! - ... + n! (n ≤ 15). Використовуйте рекурсивний метод обчислення n!

3. Прилади, устаткування та інструменти

Для виконання лабораторної роботи використовується ПЕОМ з установленим пакетом Sun Microsystems JDK 1.5 і вище та інтегрованим середовищем розробки BlueJ або Eclipse. Для написання програми на Java може бути використаний будь-який текстовий редактор, наприклад, Notepad, WordPad в MS Windows і ін.

[*] Варіант підвищеної важкості

4. Правила техніки безпеки та охорони праці

Правила техніки безпеки при виконанні лабораторної роботи регламентуються «Правилами техніки безпеки при роботі в комп'ютерній лабораторії».

5. Порядок проведення лабораторної роботи

Для виконання роботи кожен студент повинен:

1. Відповісти на контрольні питання та пройти усне опитування за теоретичним матеріалом лабораторної роботи;
2. Пройти інструктаж за правилами охорони праці;
3. Запустити на комп'ютері інтегроване середовище розробки BlueJ;
4. Отримати варіант завдання у викладача;
5. Скласти алгоритм розв’язання задачі;
6. Записати код програми на комп’ютері;
7. Відкомпілювати програму та виправити всі помилки;
8. Запустити програму на виконання;
9. Отримати результати роботи програми і показати їх викладачу;
10. Підготувати і захистити звіт до лабораторної роботи.

6. Оформлення і захист звіту

Підготовлений до захисту звіт до лабораторної роботи повинен містити:

1. титульний лист, де вказані номер і назва лабораторної роботи, відомості про виконавця;
2. номер варіанта роботи та текст завдання;
3. відповіді на контрольні запитання до лабораторної роботи;
4. текст програми алгоритмічною мовою Java;
5. лістинг результатів виконання програми.

Лабораторна робота № 3

Тема: «Передача в методи посилань на масиви. Методи класу Arrays»

1. Мета роботи

Отримання навичок обробки масивів за допомогою методів класу Arrays та розробка власних методів.

2. Завдання до лабораторної роботи

Використовуючи алгоритми, розглянуті на практичному занятті, скласти програму розрахунку заданих величин.

Зауваження: у програмі там, де це можливо треба використати методи класу Arrays та метод класу Random з пакету java.util, а також метод arraycopy() класу System.

2.1 Методичні вказівки

■ Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних тем лекційного курсу:

- Заповнення, порівняння і копіювання масивів. Порівняння об'єктів..
- Сортування масиву. Пошук у відсортованому масиві.
- Генератор випадкових чисел. Використання методів класу Arrays при роботі із двовимірними масивами

Тому під час підготовки до лабораторної роботи рекомендується повторити зазначені розділи дисципліни.

Наведемо нижче перелік важливих методів, що дозволяють полегшити обробку масивів і можуть бути використані при створенні програми лабораторної роботи.

■ **Методи заповнення масиву**

Методи знаходяться в пакеті java.util класу Arrays. Він має вісімнадцять перевантажених варіантів.

Дев'ять методів мають вигляд

```
static void fill(type[], type value)
```

де value – значення, яким заповнюється масив. Може мати примітивний тип byte, short, int, long, char, float, double, boolean або тип Object.

Дев'ять методів дозволяють заповнювати масив значенням між двома заданими індексами і також використовується з вказаними вище типами даних

```
static void fill(type[], int from, int to, type value)
```

■ **Методи порівняння масивів**

У класі Arrays є дев’ять методів equals() для порівняння масисів типа byte, short, int, long, char, float, double, boolean або типу Object.

```
static boolean equals(type[] al, type[] a2)
```

При порівнянні масивів перевіряються наступні умови: у масивах повинна бути рівна кількість елементів і кожен елемент повинен бути еквівалентний відповідному елементу іншого масиву.

■ **Методи копіювання масивів**

У класі System пакета java.lang є статичний метод копіювання масивів System.arraycopy(). Метод може використовуватися зі всіма примітивними типами або типом Object.

```
static void arraycopy(Object src, int src_ind, Object dest,
                      int dest_ind, int count)
```

З масиву, на який вказує посилання src, копіюється count елементів, починаючи з елементу з індексом src_ind, у масив, на який вказує посилання dest, починаючи з його елементу з індексом dest_ind.

■ **Методи сортування масивів**

У класі Arrays існують статичні методи для сортування масиву з різними типами числових елементів у порядку зростання чисел.

Вісім з них мають простий вигляд

```
static void sort(type[] a)
```

де type може бути одним з семи примітивних типів byte, short, int, long, char, float, double або типом Object.

Вісім методів сортують частину масиву від індексу from включно до індексу to виключно:

```
static void sort(type[] a, int from, int to)
```

■ **Методи бінарного пошуку елементу у відсортованому масиві**

Якщо масив є відсортованим, то в ньому можна швидко знайти певний елемент, використовуючи для цього методи Arrays.binarySearch().

Методи пошуку повертають індекс знайденого елементу масиву. Якщо елемент не знайдений, то повертається від'ємне число, яке означає індекс, з яким елемент був би вставлений в масив в заданому порядку, із зворотнім знаком.

Методи мають вигляд

```
static int binarySearch(type[] a, type element)
```

де type може бути типом byte, short, int, long, char, float, double, boolean.

■ **Отримання випадкових чисел**

Роботу з випадковими числами можна організувати за допомогою методів класу Random пакету java.util.

Спочатку потрібно створити генератор одним з двох способів:

Random (long n) — створює генератор псевдовипадкових чисел, який використовує для початку роботи число n; Зауваження: якщо для ініціалізації Random використовувати одне і те ж число n, буде одержана та ж сама послідовність псевдовипадкових чисел.

Random() — вибирає в якості початкового значення поточний час;

Після створення генератора, можна одержувати випадкові числа відповідного типа методами nextBoolean(), nextDouble(), nextFloat(), nextInt(), nextLong(), nextInt(int max).

Дійсні випадкові числа рівномірно розташовуються в діапазоні від 0.0 включно до 1.0 виключно.

Цілі випадкові числа рівномірно розподіляються по всьому діапазону відповідного типа за одним виключенням: якщо в аргументі вказане ціле число max, то діапазон випадкових чисел буде від нуля включно до max виключно.

2.2 Приклад програми

Задача. Знайти в двовимірному цілому масиві n×m елементів кількість елементів за модулем більших 10 в кожному рядку матриці. Рядки, в яких немає таких елементів упорядкувати за зростанням.

Розробити і використати в програмі метод заповнення матриці випадковими двозначними числами, метод виведення матриці на екран, метод, що визначає кількість елементів за модулем більших 10 в одновимірному масиві.

```
import java.util.*;
class Zadanie{

static void fill(int [][] a){
Random ran=new Random();
for(int i=0;i<a.length;i++)
for(int j=0;j<a[i].length;j++)
a[i][j]=ran.nextInt(12);}

static void print(int[][] a){
for(int i=0;i<a.length;i++) {
for(int j=0;j<a[i].length;j++)
System.out.printf("%8d",a[i][j]);
System.out.println();}}

static int count(int[] t){
int k=0;
for(int i=0;i<t.length;i++)
if (Math.abs(t[i])>10) k++;
return k;}

public static void main(String[] args){
int n=5;
int m=6;
int[][] arr=new int[n][m];
fill(arr);
int[] kol=new int[n];
System.out.println("Вихідна матриця");
print(arr);
for(int i=0;i<arr.length;i++){
kol[i]=count(arr[i]);
if (kol[i]==0) Arrays.sort(arr[i]);}
System.out.println("Нова матриця");
print(arr);
System.out.println("Кількість");
for(int i=0;i<arr.length;i++)
System.out.printf("У%2d рядку-%d%n", i+1, kol[i]);
}}
```

```
Вихідна матриця
8    5    6    11   2    0
8    9    0    8    2    8
11   2    1    4    7    11
7    11   7    2    3    4
7    2    1    11   1    2
Нова матриця
8    5    6    11   2    0
0    2    8    8    8    9
11   2    1    4    7    11
7    11   7    2    3    4
7    2    1    11   1    2
```

```
Кількість
У 1 рядку-1
У 2 рядку-0
У 3 рядку-2
У 4 рядку-1
У 5 рядку-1
```

2.3 Контрольні питання

1. Дайте визначення передачі параметрів в метод за значенням і за посиланням. Як відбувається передача параметрів в Java?
2. Як організувати генератор випадкових чисел за допомогою методів класу Random? Приведіть приклади.
3. Для чого призначений метод System.arraycopy()? Приведіть приклади його використання.
4. Опишіть призначення і формати запису методів класу Arrays, призначених для порівняння, сортування і заповнення масивів.
5. Як організувати бінарний пошук у відсортованому масиві за допомогою методів класу Arrays? Приведіть приклади.

2.4 Варіанти завдань

1. Дана ціла прямокутна матриця **n**×**m** елементів. У матриці кожен рядок необхідно упорядкувати за зростанням між мінімальним і максимальним елементом, якщо номер мінімального менше номера максимального елементу і за убуванням у протилежному випадку.

Розробити і використати в програмі метод заповнення матриці випадковими двозначними числами, метод виведення матриці на екран, методи, що визначають номери мінімального і максимального елементів в одновимірному масиві, а також метод сортування одновимірного масиву за убуванням між двома заданими індексами.

2. Дана ціла прямокутна матриця **n**×**m** елементів. У матриці кожен рядок упорядкувати або за зростанням, якщо вона містить нульові елементи, або за убуванням – в протилежному випадку.

Розробити і використати в програмі метод заповнення матриці випадковими числами від 0 до 10, метод виведення матриці на екран, логічний метод, що визначає наявність в рядку нульових елементів, а також метод сортування одновимірного масиву за убуванням.

3. Дана дійсна квадратна матриця **n×n** елементів. Побудувати послідовність дійсних чисел A1, A2, ..., An за наступним правилом: якщо в і - тому рядку матриці елемент, що належить головній діагоналі, від'ємний, то Ai дорівнює сумі максимального і мінімального елементів і - того рядка; інакше Ai дорівнює сумі останніх елементів і - того рядка, починаючи з першого в рядку від'ємного елементу.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -10.0 до 10.0, метод виведення матриці на екран, логічний метод, що визначає знак елемента одновимірного масиву з вказаним індексом, а також методи що визначають суму максимального і мінімального елементів одновимірного масиву і суму елементів одновимірного масиву від першого від'ємного до кінця масиву.

4. Дана дійсна квадратна матриця **n×n** елементів. Побудувати послідовність дійсних чисел A1, A2, ..., An за наступним правилом: якщо і-тий рядок матриці упорядкований за зростанням, то Ai дорівнює 1; інакше Ai дорівнює максимальному елементу рядка, якщо в ній є нульові елементи, або мінімальному якщо таких елементів немає.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -15.0 до 35.0, метод виведення матриці на екран, логічні методи, що визначають чи упорядкований за зростанням одновимірний масив і чи є в ньому нульові елементи, а також методи що визначають максимальний і мінімальний елемент одновимірного масиву.

5. Дана дійсна квадратна матриця **n×n** елементів. Сформувати дійсний вектор з елементів матриці, розташованих нижче головної діагоналі (включаючи елементи розташовані на головній діагоналі). Елементи, розташовані вище за головну діагональ, упорядкувати за убуванням.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -20.0 до 25.0, метод виведення матриці на екран, метод копіювання елементів одновимірного масиву до вказаного індексу в заданий вектор і метод упорядковування за убуванням елементів одновимірного масиву починаючи із заданого і до кінця масиву.

6. Визначити, чи є задана ціла квадратна матриця n-го порядку (n-парне) симетричної відносно головної діагоналі. Якщо матриця симетрична, то заповнити елементи вище головної діагоналі нулями, інакше заповнити елементи нижче головної діагоналі одиницями.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -30 до 40 (передбачити можливість створення симетричної матриці для тестування програми), метод виведення матриці на екран, логічний метод, що визначає симетричність матриці.

7. Дана ціла матриця n-го порядку, елементи якої змінюються в діапазоні [-1,1] . Сформувати вектор з послідовності $B_1...B_N$ такий, що B_i=1, якщо і-ий рядок утворює зростаючу послідовність; B_i= -1, якщо і-ий рядок утворює убуваючу послідовність і B_i=0 – в протилежному випадку. У матриці елементи рядків, які дорівнюють даному вектору, замінити на номер рядка, в якому вони розташовані.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -1 до 1, метод виведення матриці на екран, логічні методи, що визначають чи утворює одновимірний масив убуваючу або зростаючу послідовності.

8. Квадратна матриця, яка є симетричною відносно головної діагоналі, задана верхнім трикутником у вигляді одновимірного масиву. Відновити початкову матрицю і надрукувати за рядками.

Розробити і використати в програмі метод заповнення одновимірного масиву випадковими числами від -10 до 15, метод виведення матриці на екран, метод, що копіює елементи з одновимірного масиву у верхній трикутник матриці.

9. Елемент матриці назвемо сідловою точкою, якщо він є найменшим в своєму рядку і одночасно найбільшим в своєму стовпці або, навпаки, є найбільшим в своєму рядку і найменшим в своєму стовпці. Для заданої цілої матриці розміром n x m надрукувати індекси всіх її сідлових точок або вивести повідомлення, що таких точок немає. Рядки матриці, що містять сідлові точки упорядкувати за зростанням.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -5 до 35, метод виведення матриці на екран, методи, що визначають значення мінімального і максимального елементів в одновимірному масиві.

10. Дана дійсна матриця розміру n×m. Визначити кількість “особливих” елементів в рядках матриці. Елемент вважається особливим, якщо він більше суми решти елементів рядка. Надрукувати індекси всіх “особливих” точок матриці або вивести повідомлення, що таких точок немає. З “особливих” точок матриці скласти вектор, упорядкувати його за зростанням і знайти найбільше і найменше значення серед “особливих” точок.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -10 до 15, метод виведення матриці на екран, метод, який визначає кількість “особливих” точок в одновимірному масиві і метод, який виводить їх індекси на екран (у методі передбачити запис знайдених особливих точок у вектор).

11. Дана ціла матриця розміром n×m. На її основі потрібно побудувати нову матрицю таким чином: кожен парний рядок нової матриці заповнюється максимальним значенням з відповідного рядка “початкової” матриці, кожен непарний рядок нової матриці заповнюється мінімальним значенням з відповідного рядка “початкової” матриці. Слід надрукувати “початкову” і нову матриці і визначити, чи є нова матриця відсортованою, тобто всі її стовпці упорядковані за зростанням. (Вказівка: у програмі можна використовувати вектор, що є одним стовпцем нового двовимірного масиву).

Розробити і використати в програмі метод заповнення матриці випадковими числами від -25 до 35, метод виведення матриці на екран, методи, що визначають мінімальний і максимальний елементи в одновимірному масиві і логічний метод, що визначає чи утворює одновимірний масив зростаючу послідовність.

12. Дана ціла матриця розміром n×m. У матриці є “квадрат К”, якщо є підмасив розміром 2x2, що містить значення К. Наприклад, в приведеній матриці є “квадрат 9”.

1	3	2	2	8
2	9	9	6	1
12	9	9	1	4
17	6	8	5	2
1	3	5	7	1

Знайдіть в початковій матриці найбільше число К з тих, для яких в матриці існують “квадрати”. Виведіть індекси всіх “квадратів” матриці, у випадку, якщо в матриці немає жодного “квадрата”, видайте відповідне повідомлення. Визначте, чи є серед рядків матриці однакові рядки. Виведіть їх номери.

Розробити і використати в програмі метод заповнення матриці випадковими числами від 1 до 5, метод виведення матриці на друк, логічний метод, що визначає наявність “квадрата” за введеному індексу рядка і стовпця елементу.

13. Для заданої цілої матриці **A** розміром 20×20 побудувати вектор **B** довжиною 20 такий, що $\mathbf{B_i}$ є сумою елементів головної і побічної діагоналей, що знаходяться нижче **i**-го рядка. У матриці А всі рядки з парними номерами замінити вектором В.

Розробити і використати в програмі метод заповнення матриці випадковими числами від -10 до 10, метод виведення матриці на друк, метод, що визначає суму елементів головної і побічної діагоналей, що знаходяться нижче **i**-го рядка (**i** передається як параметр).

3. Прилади, устаткування та інструменти

Для виконання лабораторної роботи використовується ПЕОМ з установленим пакетом Sun Microsystems JDK 1.5 і вище та інтегрованим

середовищем розробки BlueJ або Eclipse. Для написання програми на Java може бути використаний будь-який текстовий редактор, наприклад, Notepad, WordPad в MS Windows і ін.

4. Правила техніки безпеки та охорони праці

Правила техніки безпеки при виконанні лабораторної роботи регламентуються «Правилами техніки безпеки при роботі в комп'ютерній лабораторії».

5. Порядок проведення лабораторної роботи

Для виконання роботи кожен студент повинен:

1. Відповісти на контрольні питання та пройти усне опитування за теоретичним матеріалом лабораторної роботи;
2. Пройти інструктаж за правилами охорони праці;
3. Запустити на комп'ютері інтегроване середовище розробки BlueJ;
4. Отримати варіант завдання у викладача;
5. Скласти алгоритм розв’язання задачі;
6. Записати код програми на комп’ютері;
7. Відкомпілювати програму та виправити всі помилки;
8. Отримати результати роботи програми і показати їх викладачу;
9. Підготувати і захистити звіт до лабораторної роботи.

6. Оформлення і захист звіту

Підготовлений до захисту звіт до лабораторної роботи повинен містити:

1. титульний лист, де вказані номер і назва лабораторної роботи, відомості про виконавця;
2. номер варіанта роботи та текст завдання;
3. відповіді на контрольні запитання до лабораторної роботи;
4. текст програми алгоритмічною мовою Java;
5. лістинг результатів виконання програми.

Лабораторна робота № 4

Тема: «Файлове введення/виведення. Рядки. Робота з методами класу String і StringBuffer»

1. Мета роботи

Отримання практичних навичок роботи з текстовими файлами та методами класів String, StringBuffer і StringTokenizer.

2. Завдання до лабораторної роботи

Використовуючи алгоритми, розглянуті на практичному занятті, скласти програму згідно отриманого варіанта.

Зауваження: у програмі ім'я текстового файлу повинно бути введено як параметр виклику методу main().

2.1 Методичні вказівки

■ Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних тем лекційного курсу:

- Поняття конструктора. Операція new.
- Консольне введення - виведення.
- Робота з рядками. Приклади рішення завдань із використанням алгоритмів обробки рядків.
- Методи класа String.
- Методи класа StringBuffer і StringTokenizer.

Тому під час підготовки до лабораторної роботи рекомендується повторити зазначені розділи дисципліни.

■ **Конструктор** – це спеціальний метод, який викликається інтерпретатором Java завжди, коли оператором new створюється черговий екземпляр класу. Конструктор завжди носить ім'я класу.

Перерахуємо деякі особливості конструктора.

- Конструктор є в будь-якому класі. Навіть якщо ви його не написали, компілятор Java сам створить конструктор за умовчанням (default constructor). Проте, потрібно пам'ятати, що якщо ви вже визначили деякий конструктор або декілька (з аргументами або без), то компілятор більше не буде створювати конструктор за умовчанням.
- Конструктор виконується автоматично при створенні екземпляра класу, після розподілу пам'яті, але до початку використання об'єкту.

- Конструктор не повертає ніякого значення. Тому в його описі не пишуть навіть слово void, але можна задати один з трьох модифікаторів public, protected або private.
- Конструктор не є методом, він навіть не вважається членом класу.
- У класі може бути декілька конструкторів. Оскільки у них одне і те ж ім'я, яке співпадає з ім'ям класу, то вони повинні відрізнятися типом і/або кількістю параметрів.

■ **Введення з консолі**

Базові класи для введення/виведення розташовані в пакеті java.io. Тому для роботи з ними потрібно записати import java.io.*, а в програмі оголосити екземпляри відповідних класів.

```
InputStreamReader is = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(is);
```

Для введення даних типу String слід використовувати метод readLine() класу BufferedReader.

При введенні числових даних доведеться проводити перетворення типу String у відповідний тип. Для цього використовуються наступні методи:

```
Integer.valueOf(numberStr).intValue();
Long.valueOf(numberStr).longValue();
Float.valueOf(numberStr).floatValue();
Double.valueOf(numberStr).doubleValue();
```

Класи-оболонки Integer, Short, Long, Float і Double, розташовані в пакеті java.lang, містять також статичні методи

```
parseInt(String s);
parseShort(String s);
parseLong(String s);
parseFloat(String s);
parseDouble(String s);
```

які повертають числові еквіваленти рядка s.

Розглянемо приклад введення з клавіатури значення цілої змінної.

```
import java.io.*;
class Vvod{
public static void main (String[] args) throws Exception{
int a;
BufferedReader br = new
        BufferedReader(new InputStreamReader(System.in));
System.out.println("Введіть число a:");
int a = Integer.parseInt(br.readLine());
System.out.println(a); }}
```

■ Файлове введення/виведення

Розглянемо організацію файлового потоку за допомогою класів FileReader (для читання з файлу) і FileWriter (для запису у файл).

У конструкторах файлового потоку задається ім'я файлу у вигляді рядка типу String. Конструктори не тільки створюють об'єкт, але і відшукують та відкривають файл. Наприклад:

```
String fileName = "c:\\Vixod.txt";
FileReader fr = new FileReader(fileName);
```

Після відкриття вихідного потоку типу FileWriter вміст файлу, якщо він був не порожній, стирається. Для того, щоб можна було робити дозапис в кінець файлу, в класі передбачений конструктор з двома аргументами. Якщо другий аргумент рівний true, то відбувається дозапис в кінець файлу, якщо false, то файл заповнюється новою інформацією. Наприклад:

```
FileWriter fw = new FileWriter("c:\\Vixod.txt", true);
```

Пам’ятайте, що вміст файлу, відкритого на запис конструктором з одним аргументом, стирається.

Класи файлового введення/виведення не займаються буферизацією. Для цієї мети служать класи

```
BufferedReader br = new BufferedReader(fr);
BufferedWriter bw = new BufferedWriter(fw);
```

Потоки fr і fw були визначені вище. Вони повинні приєднуватися до потоків введення/виведення.

Відразу після виконання конструктора можна читати файл методом br.readLine(); або записувати в нього методом bw.write();

Для того, щоб виведення в текстовий файл можна було організовувати за допомогою методів print(), println() і printf() слід використовувати клас PrintWriter pw=new PrintWriter(bw);

Після закінчення роботи з файлом потік слід закрити методом close().

Розглянемо приклад виведення символьних даних у текстовий файл.

```
import java.io.*;
class Input{
public static void main(String[] args) {
String fileName = "c:\\file.txt";
// Рядок, який буде записаний у файл
String data = " Рядок для запису";
try{
FileWriter fw = new FileWriter(fileName);
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter pw=new PrintWriter(bw);
```

```
// 3 рази запишемо рядок у файл
for(int i=3;--i>=0;) pw.println(data);
bw.close();

FileReader fr = new FileReader(fileName);
BufferedReader br = new BufferedReader(fr);
String s = null;
// Читаємо дані з файлу, відображаючи їх на екран
while((s=br.readLine())!=null)
System.out.println(s);
br.close();
}catch(Exception e){ }     }}
```

Для зчитування даних з текстового файлу також можна використовувати клас **Scanner**. Зазвичай це роблять за допомогою логічних методів класу, наприклад, методу **hasNext()**, який повертає true, якщо в потоці є ще дані для зчитування, а також інших варіантів логічних методів для контролю доступності зчитування даних різних типів (**hasNextDouble()**, **hasNextInt(), hasNextBoolean()** та ін.)

Розглянемо приклад використання класу Scanner для розрахунку середнього арифметичного чисел, що зчитуються з текстового файлу.

```
import java.util.*;
import java.io.*;
class Zadanie{
public static void main(String[] args)throws IOException {
int count = 0;
double sum =0.;
//Запис в текстовий файл
String name = "test.txt";
FileWriter fout = new FileWriter(name);
fout.write("2 3.4 5 6 7.4 9.1 10.5");
fout.close();
//Зміна локалі з руської на англійську
Locale newLocale = new Locale ("en", "GB");
Locale.setDefault(newLocale);
FileReader fin = new FileReader(name);
Scanner src = new Scanner (fin);
//Зчитуємо дані та розраховуємо суму
while(src.hasNext()) {
if(src.hasNextDouble()) {
sum+=src.nextDouble();
count++;
}}
fin.close();
System.out.printf("Average is %5.3f",sum/count);
}}
```

Можливо також використовувати клас Scanner для зчитування з файлу даних різних типів, навіть коли порядок їх слідкування наперед невідомий. Треба просто перевіряти тип даних, що доступні для зчитування, перед тим як зчитати їх.

■ Клас String

Клас String є основним класом, призначеним для зберігання і обробки рядків символів.

Для створення екземплярів класу String може бути використаний один з наступних конструкторів:

```
String ()
String (String str)
String (StringBuffer strbuf)
String (char[] arr)
String (char[] arr, int first, int count)
```

Перший з них створює порожній рядок, другий і третій копіюють вміст об'єктів класів String і StringBuffer в створений об'єкт. Останні два конструктори дозволяють створити рядок на основі символьного масиву або його частини. Крім того, будь-яке об'єктне посилання типу String може бути проініціалізоване за допомогою привласнення йому рядкового літерала. Наприклад,

```
String filename = "data.txt";
```

■ Основні методи класу String

Метод	**Опис**
int length() **char** charAt(**int** index) **char**[] toCharArray()	Отримання довжини рядка Пошук символу Отримання рядка у вигляді символьного масиву
boolean equals(String str) **boolean** equalsIgnoreCase(String str) **boolean** startsWith(String prefix) **boolean** endsWith(String suffix)	Порівняння рядків на рівність Порівняння рядків без урахування регістру Перевірка, чи починається рядок із заданого підрядка Перевірка, чи закінчується рядок заданим підрядком

Метод	Опис
int indexOf(String subStr) **int** indexOf(String subStr, **int** fromIndex) **int** lastIndexOf(String subStr) **int** lastIndexOf(String subStr, **int** fromIndex)	Пошук першого входження підрядка в рядок з початку рядка Теж саме, але із заданої позиції Пошук останнього входження підрядка в рядок з початку рядка Теж саме, але із заданої позиції
String substring(**int** beginIndex, **int** endIndex) String substring(**int** beginIndex) String concat(String str) String toUpperCase() String toLowerCase() String trim() String replace(String target, String replacement)	Отримання підрядка (символ endIndex не входить до підрядка!) Отримання хвоста рядка Конкатенація рядків Перетворення рядка до верхнього/ нижнього регістру Видалення провідних і завершуючих пробілів в рядку Заміна підрядка іншим рядком

■ **Клас StringBuffer**

Клас StringBuffer призначений для роботи з рядками, що модифікуються. Кожен об'єкт цього класу містить деякий буфер, що зберігає рядок. При змінах рядка розмір буфера може автоматично змінюватися.

Для створення екземплярів класу StringBuffer використовуються наступні конструктори.

```
StringBuffer ()
StringBuffer (int capacity)
StringBuffer (String str)
```

Перші два з них створюють об'єкт, що зберігає порожній рядок з початковим буфером довжини 16 в першому випадку і заданої довжини в другому випадку. Третій конструктор ініціалізує буфер заданим рядком. Розмір буфера в цьому випадку встановлюється рівним довжині рядка, збільшеного на 16.

■ **Основні методи класу StringBuffer**

Метод	**Опис**
int length() **char** charAt(**int** index) **char** setCharAt(**int** index)	Отримання довжини рядка Пошук символу Зміна символу
StringBuffer append(String str) StringBuffer append(**int** i) StringBuffer append(Object obj) . . .	Додавання рядка до кінця (існують перевантажені версії для всіх примітивних типів і типу Object)
StringBuffer insert(**int** index, String str)	Вставка рядка в задану позицію
StringBuffer delete(**int** beginIndex, **int** endIndex)	Видалення підрядка
StringBuffer replace(**int** beginIndex, **int** endIndex, String str)	Заміна одного підрядка іншим
StringBuffer reverse()	Звернення рядка

■ **Клас StringTokenizer**

У пакет java.util входить простий клас StringTokenizer, що полегшує синтаксичний розбір рядків. У ньому три конструктори.

```
StringTokenizer (String str)
StringTokenizer (String str, String delimeters)
StringTokenizer(String str, String delimeters, boolean
flag)
```

Перший конструктор створює об'єкт, готовий розбити рядок на слова, які розділені пробілами, символами табуляції '\t', переводу рядка '\n' і повернення каретки '\r' . Роздільники не включаються до числа слів. Другий конструктор задає роздільники другим параметром delimeters, наприклад:

```
StringTokenizer("Казнить,нельзя:пробелов-нет", " \t\n\r,:-");
```

Тут перший роздільник — пробіл. Потім йдуть символ табуляції, символ переводу рядка, символ повернення каретки, кома, двокрапка, дефіс. Порядок розташування роздільників в рядку delimeters не має значення. Роздільники не включаються до числа слів. Третій конструктор дозволяє включити роздільники до числа слів. Якщо параметр flag рівний true, то роздільники включаються до числа слів, якщо false — ні.

■ **Основні методи класу StringTokenizer**

Метод	**Опис**
String nextToken ()	Повертає у вигляді рядка наступне слово
boolean hasMoreTokens ()	Повертає true, якщо в рядку ще є слова, і false, якщо слів більше немає
int countTokens ()	Повертає число слів, що залишилися
String nextToken(String newDelimeters)	Дозволяє під час виконання програми міняти роздільники. Наступне слово буде виділено згідно нових роздільників newDelimeters

Приклад розбиття рядка на слова:

```
String s = "Рядок, який ми хочемо разібрати на слова";
StringTokenizer st = new StringTokenizer(s, " \t\n\r,.");
while(st.hasMoreTokens()){
// Отримуємо слово і виводимо на екран
System.out.println(st.nextToken());  }
```

2.2 Приклад програми

Прочитати рядки з текстового файлу і поміняти місцями перше і останнє слово в кожному рядку, новий текст записати в той же файл.

```
import java.io.*;
class Primer{
public static void main(String[] args){
String fileName=args[0];
try{
FileReader fr = new FileReader(fileName);
BufferedReader br = new BufferedReader(fr);
FileWriter fw = new FileWriter(fileName,true);;
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter pw = new PrintWriter(bw);
String s = null;
int count = 0;
pw.println();
while((s=br.readLine())!=null){
int nom1=s.indexOf(' ');
String s1=s.substring(0,nom1+1);// Перше слово
int nom2=s.lastIndexOf(' ');
```

```
String s2=s.substring(nom2,s.length());//Останнє слово
String s3=s.substring(nom1,nom2+1);
s=s2+s3+s1;
pw.println(s);}
br.close(); pw.close();
}catch(Exception e){
e.printStackTrace();}
}}
```

Другий варіант програми з використанням методів класу StringTokenizer

```
import java.io.*;
import java.util.*;
class Primer{
public static void main(String[] args){
String fileName=args[0];
try{
FileReader fr = new FileReader(fileName);
BufferedReader br = new BufferedReader(fr);
FileWriter fw = new FileWriter(fileName,true);;
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter pw = new PrintWriter(bw);

String s = null,s1=null,s2=null,s3=null;
int count = 0;
pw.println();
while((s=br.readLine())!=null){
s2="";
StringTokenizer st = new StringTokenizer(s, " \t\n\r,.");
int n=0;
while(st.hasMoreTokens()){
if (n==0) s1=st.nextToken(); //Перше слово
else
{if (st.countTokens()==1) s3=st.nextToken();//Останнє слово
else s2+=st.nextToken();}
n++;}
s=s3+" "+s2+" "+s1;
pw.println(s);}
br.close(); pw.close();
}catch(Exception e){
e.printStackTrace();   }}}
```

2.3 Контрольні питання

1. За допомогою яких методів можна перетворити в рядок змінні примітивних типів і масив символів? Як перетворити з String в примітивний тип? Приведіть приклади.

2. Які методи класу String дозволяють провести заміну регістра букв?
3. У яких випадках використовується клас StringBuffer? Як можна встановити ємність буфера для об'єкту класу StringBuffer? Опишіть конструктори даного класу.
4. Дайте короткий опис конструкторам і методам класу StringTokenizer.
5. У чому відмінність між наступними фрагментами коду:

```
1. String s1 = "abc";
String s2 = new String("abc");
boolean result = (s1==s2);

2. String s1 = new String("abc");
String s2 = new String("abc");
boolean result = (s1.equals(s2));
```

6. Як в Java можна організувати файлові потоки введення-виведення?

2.4 Варіанти завдань

1. Прочитати текст Java-програми і всі слова "public" в оголошенні атрибутів класу замінити на "private". Новий текст записати в той же файл.

2. Прочитати текст Java-програми і записати в інший файл у зворотному порядку символи кожного рядка.

3. Прочитати текст Java-програми і в кожному слові коротше за п'ять символів всі символи замінити знаком *. Результат записати в інший файл.

4. Напишіть програму для проведення пошуку і заміни слів в тексті. Вона повинна приймати три параметри виклику методу main(): ім'я файлу, рядок для пошуку і рядок для заміни. В результаті її роботи створюється файл з ім'ям res.txt, що містить результати заміни.

5. Прочитати текст Java-програми і видаліть з нього всі слова що починаються з символу ‘p’. Новий текст виведіть в інший файл.

6. Прочитати текст Java-програми і в кожному слові довше за два символи всі рядкові символи замінити прописними. Результат записати в інший файл.

7. З файлу видалити всі слова, що містять від трьох до п'яти символів, але при цьому з кожного рядка треба видалити тільки максимальну парну кількість таких слів.

8. У файлі, що містить в кожному окремому рядку прізвища студентів і їх оцінки, записати великими буквами прізвища тих студентів, які мають середній бал більше "4". Приклад файлу:
Петренко 5 4 5 3 3
Шевченко 5 5 4 3

9. Прочитати текст Java-програми і визначити, скільки в ньому оголошених класів. Видалити в оголошенні класів модифікатор "public". Результат вивести в той же файл.

10. З тексту Java-програми видалити коментарі виду // і /* */. Новий текст записати в той же файл.

11. Створити і заповнити файл випадковими цілими числами. Відсортувати вміст файлу за збільшенням і результати вивести в цей же файл.

12. Прочитати текст Java-програми і видалити з нього всі "зайві" пробіли, залишивши тільки значущі.

13. Прочитати текст Java-програми і у вихідний файл записати всі статично виділені в програмі змінні типу int і double в наступному форматі:
Змінні типу int: a,b,c.
Змінні типу double: x,y,z.

14. Прочитати текст Java-програми і у вихідний файл записати всі цикли **for** разом з їх вмістом (у відповідності з синтаксисом циклу for). Цикли розділяти порожнім рядком. **Вказівка:** врахувати можливість існування вкладених циклів.

15. Прочитати текст Java-програми і у кожному рядку всі слова, довжина яких перевищує середню довжину слів в рядку, скоротити до середньої довжини. Результат записати в інший файл.

3 Прилади, устаткування та інструменти

Для виконання лабораторної роботи використовується ПЕОМ з установленим пакетом Sun Microsystems JDK 1.5 і вище та інтегрованим середовищем розробки BlueJ або Eclipse. Для написання програми на Java може бути використаний будь-який текстовий редактор, наприклад, Notepad, WordPad в MS Windows і ін.

4 Правила техніки безпеки та охорони праці

Правила техніки безпеки при виконанні лабораторної роботи регламентуються «Правилами техніки безпеки при роботі в комп'ютерній лабораторії».

5 Порядок проведення лабораторної роботи

Для виконання роботи кожен студент повинен:

1. Відповісти на контрольні питання та пройти усне опитування за теоретичним матеріалом лабораторної роботи;
2. Пройти інструктаж за правилами охорони праці;
3. Запустити на комп'ютері інтегроване середовище розробки BlueJ;
4. Отримати варіант завдання у викладача;
5. Скласти алгоритм розв’язання задачі;
6. Записати код програми на комп’ютері;
7. Відкомпілювати програму та виправити всі помилки;
8. Запустити програму на виконання;
9. Отримати результати роботи програми і показати їх викладачу;
10. Підготувати і захистити звіт до лабораторної роботи.

6 Оформлення і захист звіту

Підготовлений до захисту звіт до лабораторної роботи повинен містити:

1. титульний лист, де вказані номер і назва лабораторної роботи, відомості
   про виконавця;
2. номер варіанта роботи та текст завдання;
3. відповіді на контрольні запитання до лабораторної роботи;
4. текст програми алгоритмічною мовою Java;
5. лістинг результатів виконання програми.

Лабораторна робота № 5

Тема: «Динамічні структури даних. Списки, черги, стеки, бінарні дерева та алгоритми їх оброблення»

1. Мета роботи

Отримання практичних навичок роботи з динамічними типами даних в Java: списками, чергами, стеками і бінарними деревами та алгоритмами їх оброблення.

2. Завдання до лабораторної роботи

Використовуючи алгоритми, розглянуті на практичному занятті, скласти програму згідно отриманого варіанта.

Завдання складається з двох частин. У першій частині слід використовувати стеки, черги чи списки. У другій частині необхідно скласти програму побудови непустого бінарного дерева з вказаним типом елементів дерева (ТЕД) і написати метод, який виконує вказані для свого варіанту дії з елементами дерева.

2.1 Методичні вказівки

■ Лабораторна робота спирається на знання й уміння, отримані при вивченні наступних тем лекційного курсу:

– Поняття конструктора. Операція new;

– Абстрактний тип даних – стек.

– Абстрактний тип даних – черга;

– Абстрактний тип даних – список;

– Бінарні дерева та алгоритми їх оброблення.

Тому під час підготовки до лабораторної роботи рекомендується повторити зазначені розділи дисципліни.

■ **Стек** являє собою об'єкт, робота з яким – додавання і видалення – здійснюється за принципом **LIFO** (last-in first-out – останнім прийшов – першим пішов). Додавання об'єктів в стек може здійснюватися в будь-який момент, видаляється ж лише об'єкт, доданий останнім.

Стек S є абстрактним типом даних (АТД), який підтримує наступні основні методи:

push (obj): поміщає об'єкт obj на вершину стека.

pop (): видаляє об'єкт з стека і повертає новий верхній об'єкт стека; якщо стек пуст, видається повідомлення про помилку.

size (): повертає кількість об'єктів в стеці.

isEmpty (): повертає логічне значення true, якщо стек пуст.

peek (): повертає верхній об'єкт в стеці не видаляє його; якщо стек пуст, видається повідомлення про помилку.

Приклад реалізації стека за допомогою масиву

```
public class ArrayStack{
    private Object S[];
    private int t;

    public ArrayStack(int capacity){
        S = new Object[capacity];
        t = -1; //індекс останього елементу в стеці
    }
    public int size() {
    return (t + 1);
    }
    public boolean isEmpty(){
            return (t == -1);
    }
    public void push(Object obj){
        if(t+1 < S.length)
            S[++t] = obj;
    }
    public Object pop(){
        if(isEmpty())
            return null;
        return S[t--];
    }
    public Object peek(){
            if(isEmpty())
            return null;
        return S[t];
    }
    // метод поиска элемента в стеке
    public int search(Object obj){
            int n=-1;
            if(isEmpty()) return -1;
            for(int i=1; i<S.length; i++)
            {if (S[i].equals(obj)) n=i; break;}
            return n;
        }}
```

Елементами такої реалізації стека можуть бути дані будь-якого типу.

Клас Stack

В Java існує «вбудований» клас Stack, який належить пакету java.util. Розглянемо конструктор и методи цього класу.

Конструктор

Stack()

Створює пустий стек.

Методи	
boolean	empty() Служить для перевірки стека на наявність елементів –він повертає true, якщо стек порожній.
Object	peek() Повертає верхній елемент, не видаляючи його з стека.
Object	pop() Дістає верхній елемент, видаляючи його з стека.
Object	push(Object item) Поміщає елемент у вершину стека.
int	search(Object obj) Метод шукає заданий елемент у стеку, повертаючи кількість операцій pop, які потрібні для того щоб перевести шуканий елемент у вершину стека. Якщо заданий елемент у стеку відсутній, цей метод повертає -1.

■ **Черга** – ще одна базова структура даних, що об’єднує об’єкти, робота з якими – додавання та видалення – здійснюється за принципом ***FIFO (first-in first-out)*** («першим прийшов – першим пішов»). Додавання об’єктів в чергу може здійснюватися в будь який час, але видалити можна лише об’єкт, який був доданий першим. Тобто, елементи додаються в чергу з кінця, а видаляються з початку.

Черга Q – абстрактний тип даних (АТД), який підтримує наступні основні методи:

enqueue (obj): поміщає об’єкт в кінець черги.

dequeue (): проводить видалення та повертає об’єкт з початку черги; якщо черга пуста, видається повідомлення про помилку.

size (): повертає кількість об’єктів в черзі.

isEmpty (): повертає логічне значення, яке підтверджує що черга пуста.

front (): повертає перший об’єкт в черзі, не видаляючи його; якщо черга пуста, видається повідомлення про помилку.

Приклад реалізації черги на основі масиву

В програмі використовуються дві змінні – вказівника head і tail:

• head – вказівник на голову (початок черги), тобто на індекс комірки Q, в якій зберігається перший елемент черги (кандидат на видалення при виконанні метода dequeue) при умові, що черга не є пустою (в цьому випадку head = tail).

• tail – вказівник на хвіст (кінець черги), тобто індекс наступної доступної (вільної) комірки масиву Q.

Після видалення елемента з початку черги, значення head збільшується на 1 для отримання індексу наступної комірки масиву. Якщо додається новий елемент, то значення tail збільшується на 1 для отримання індексу наступної доступної комірки масиву Q.

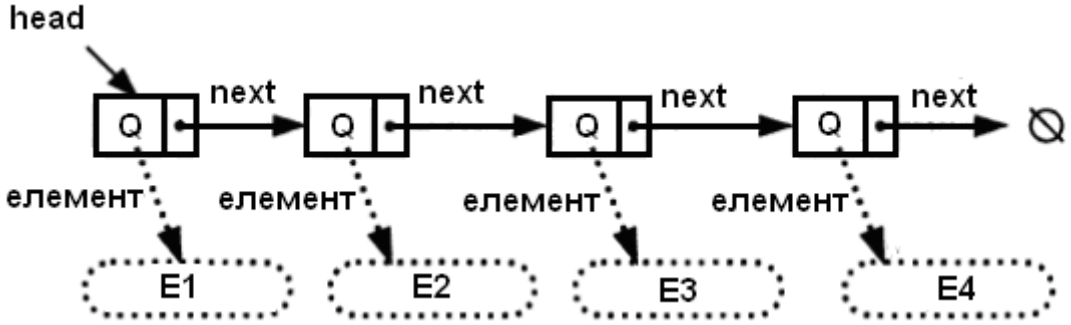
В програмній реалізації часто черга Q розглядається як «круговий» масив, який продовжується от Q[0] до Q[N- 1], а потім знову повертається до Q[0], тобто можна обнуляти значення head і tail у випадку коли в результаті їх збільшення значення перевищують розмір черги. Можна використовувати і інші способи, наприклад, при кожному збільшенні значення head і tail на 1 записувати цю операцію у вигляді «(head + 1) % N» або «(tail + 1) % N» відповідно, де % позначає операцію ділення по модулю, яка обчислюється шляхом отримання залишку від ділення. x % y =x - [x/y]·y, де y ≠ 0. Таким чином, за допомогою операції ділення по модулю вирішується проблема перевищення меж масиву.

При head = tail, черга може бути як порожньою так і повною. Для вирішення даного протиріччя можна встановити умову, що Q не може містити більше N-1 об'єктів. Можна, і це показано нижче, застосовувати додаткову логічну змінну full для ідентифікації переповнення черги.

```
public class ArrayQueue{
    private Object[] Q;
    private int head,tail;
    private boolean full;

    public ArrayQueue(int maxsize){
        Q = new Object[maxsize];
        head = tail = 0;
        full = false;
    }

    public int size(){
        return ((Q.length-head+tail)%Q.length);
    }
```

```
        public boolean isEmpty(){
        return ((head == tail) &&(!full));
    }

    public Object front() {
        if (isEmpty()) return null;
         return Q[head];
     }
     public void enqueue (Object o){
        if(!full)  Q[tail=(++tail % Q.length)] = o;
        if(head==tail) full=true;
    }

    public Object dequeue(){
       if(full) full = false;
        else
        if(isEmpty())
            return null;
        return Q[head=(++head%Q.length)];
    }
}
```

■ **Список** – форма сукупності вузлів, упорядкованих в лінійну послідовність, де кожен вузол є складовим об'єктом, що містить посилання на якийсь елемент, а також на наступний вузол (див. рисунок).



Однонаправлений зв'язний список

Посилання next всередині вузла є ***вказівником (або покажчиком)*** на наступний вузол. Таким чином, перехід за посиланням next до наступного вузла називається перемиканням за вказівником. Перший і останній вузли зв'язного списку називаються головою і хвостом списку відповідно. Хвіст списку визначимо як вузол з нульовим значенням посилання next, що означає, що список завершений. Програмісту доступний вказівник на початок списку. Знаючи його, можна отримати доступ до будь-якого елементу, пройшовши послідовно попередні елементи. На відміну від

масиву, у списку немає заздалегідь встановленої довжини, простір, що використовується їм, пропорційний числу його елементів.

Реалізація вузла однонаправленого зв'язного списку

```
public class Node {
// змінні екземпляру
private Object element;
private Node next;

// конструктори
public Node() {
this(null, null);
}
public Node(Object e, Node n) {
element = e;
next = n;
}

// аксесорні методи
Object getElement() {
return element;
}
Node getNext() {
return next;
}
// модифікуючи методи
void setElement(Object newElem) {
element = newE1em;
}
void setNext(Node newNext) {
next = newNext;
}}
```

Однак існує тип зв'язного списку, який має кращі можливості, в тому числі додавання і видалення елементів на обох кінцях списку – двоз'вязний список. Вузол двозв'язного списку містить два посилання – посилання до наступного вузла списку ***next*** і посилання до попереднього вузла списку ***prev***. Java-реалізація вузлів двозв'язного списку представлена фрагментом коду нижче.

Реалізація вузла двозв'язного списку

```
class DLNode {
private Object element;
private DLNode next, prev;

DLNode() { this(null, null, null); }
DLNode(Object e, DLNode p, DLNode n) {
element = e;
```

```
next = n;
prev = p;
}

void setElement(Object newElem) { element = newElem; }
void setNext(DLNode newNext) { next = newNext; }
void setPrev(DLNode newPrev) { prev = newPrev; }

Object getElement() { return element; }
DLNode getNext() { return next; }
DLNode getPrev() { return prev; }
}
```

Для спрощення програмування рекомендується додавати спеціальні вузли на обох кінцях списку: вузол ***first*** перед першим вузлом списку і вузол ***last*** безпосередньо після останнього вузла списку. Дані «холості», або сигнальні вузли не містять елементів. Вузол ***first*** містить дійсне посилання next і нульове посилання prev, у тій час як вузол ***last*** — дійсне посилання prev і нульове посилання next.

Програмна реалізація окремих методів АТД «список» приведена у прикладі виконання лабораторної роботи нижче (див. п.2.2).

■ ***Дерево*** – це абстрактний тип даних (АТД) для ієрархічного зберігання елементів. За винятком елемента на чолі дерева, кожний елемент структури має батька (parent element) і нуль або більше дочірніх елементів (children element). Звичайно головний елемент дерева називається коренем (root). У нього на відміну від інших вузлів немає батька. Тобто ***дерево (tree) T*** – це набір вузлів (node), що зберігають елементи, які перебувають у відносинах «батьки і діти», з наступними властивостями:

- ***T*** має особливий вузол ***r*** – корінь даного дерева (***root*** of ***T***);
- кожний вузол ***v*** цього ***T***, відмінний від ***r***, має батьківський вузол ***u***.

Бінарним деревом (binary tree) називається упорядковане дерево, в якому кожний з вузлів має максимум два дочірніх елемента. Бінарне дерево вважається ***правильним (proper)***, якщо кожний вузол не містить жодного або містить два дочірніх елемента. Тобто, в правильному бінарному дереві кожний складовий вузол має два дочірніх елемента. Дочірні елементи в таких вузлах маркуються як ***«правий» і «лівий» (left child і right child)***. І розподіляються ці елементі таким чином, що лівий стоїть перед правим. Кожний вузол бінарного дерева крім даних зберігає посилання на лівий і правий дочірній елемент, для яких він є батьком. Вузли без дочірніх елементів називаються ***листями (leaves)***. Бінарне дерево за своєю

природою рекурсивно. Дочірній елемент будь-якого вузла є коренем деякого дерева. Можна навіть говорити про ліві і праві піддерева.

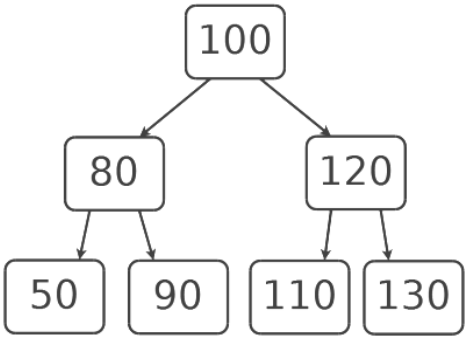
Важливою характеристикою бінарного дерева є ***глибина*** – максимальна довжина шляху від кореня до деякого вузла (очевидно що вона досягається для деякого листа). Вузли з однаковою глибиною (відстанню від коріння) утворюють шар (або рівень). Нехай, ***v*** — вузол дерева ***T***. Глибина вузла ***v*** виражається кількістю батьків ***v***, виключаючи сам ***v***.

Глибина вузла ***v*** може бути рекурсивно визначена наступним чином:

- якщо ***v*** — корінь, то його глибина дорівнює 0;
- в протилежному випадку глибина ***v*** дорівнює одиниці плюс глибина батька ***v***.

Один з частих випадків бінарних дерев – ***бінарні дерева пошуку***. Вузол бінарного дерева пошуку крім посилань на дочірні елементи повинен містити унікальний ключ. Ключи повинні бути порівнянними. Для бінарних дерев пошуку з унікальними ключами виконуються наступні властивості: ключ у корінні більше будь-якого ключа у лівому піддереві, менше будь-якого ключа у правому піддереві, ліве і праве піддерева самі є бінарними деревами пошуку.

На рисунку наведений приклад бінарного дерева пошуку для послідовності чисел:



Зазвичай для бінарного дерева вводяться операції вставки, пошуку, видалення та інш. Програмна реалізація окремих методів приведена у прикладі виконання лабораторної роботи нижче (див. п.2.2).

2.2 Приклад програми

■ Використовуючи динамічну структуру список, зберегти ряд випадкових цілих чисел. Видалити перший елемент у списку, що дорівнює 3. Видалити усі від'ємні елементи. Вивести зміст списку у зворотному порядку.

```
//Клас, що реалізує окремий вузол двозв'язного списку
class Link
{
   private int data;
   private Link next;
   private Link prev;

//Конструктори
   public Link()
   {
      data = 0;
      next = null;
      prev = null;
   }
   public Link(int value)
   {
      data = value;
      next = null;
      prev = null;
   }

   public void displayLink()
   {
       System.out.print(data + " ");
   }
}

//Клас, що містить методи роботи зі списком і вирішує
//завдання варіанту.

import java.util.*;
public class DoublyLinkedList {
  //сигнальні вузли
  private Link first;
  private Link last;

  //конструктор
  public DoublyLinkedList() {
    first = null;
    last = null;
  }

  public boolean isEmpty(){
    return first == null;
  }

  //Метод вставляє новий елемент dd в якості першого
  //елементу списку.
public void insertFirst(int dd){
    Link newLink = new Link(dd);
```

```
    if (isEmpty())
      last = newLink;
    else
      first.prev = newLink;
    newLink.next = first;
    first = newLink;
  }

//Метод вставляє новий елемент dd в якості останнього
//елемента списку.
  public void insertLast(int dd){
    Link newLink = new Link(dd);
    if (isEmpty())
      first = newLink;
    else {
      last.next = newLink;
      newLink.prev = last;
    }
    last = newLink;
  }

  //Метод видаляє перший елемент списку
  public Link deleteFirst(){
    Link temp = first;
    if (first.next == null)
      last = null;
    else
      first.next.prev = null;
    first = first.next;
    return temp;
  }

  //Метод видаляє останній елемент списку
  public Link deleteLast(){
    Link temp = last;
    if (first.next == null)
      first = null;
    else
      last.prev.next = null;
    last = last.prev;
    return temp;
  }

 //метод вставляє елемент dd після елементу key
  public boolean insertAfter(int key, int dd) {
    Link current = first;
    while (current.data != key){
      current = current.next;
      if (current == null)
       return false; // не знайдено
    }
```

```
    Link newLink = new Link(dd); // створюємо новий список

    if (current == last) //якщо останній елемент
    {
      newLink.next = null;
      last = newLink;
    } else // не останній елемент
    {
      newLink.next = current.next;

      current.next.prev = newLink;
    }
    newLink.prev = current;
    current.next = newLink;
    return true; // знайдено, додано
  }

 //метод видаляє перший елемент у списку, який дорівнює key
  public Link deleteKey(int key){
    Link current = first;
    while (current.data != key)
    {
      current = current.next;
      if (current == null)
        return null; //такий елемент не знайдений
    }

    if (current == first) //знайдений, перший вузол
      first = current.next;
    else
      current.prev.next = current.next;

    if (current == last) // останній вузол?
      last = current.prev;
    else
      // not last
      current.next.prev = current.prev;
    return current; // return value
  }

//метод видаляє у списку усі від'ємні елементи
public Link delete()
 {
    Link current = first;
    while (current != null)
    { if (current.data < 0)
    {if (current == first) //це перший елемент?
      first = current.next;
    else
      current.prev.next = current.next;
```

```
    if (current == last) //вузол останній?
      last = current.prev;
    else
      // не последний
      current.next.prev = current.prev;
    }
      current = current.next;
    }
    return current; //повертаємо новий список
  }

//Виведення елементів списку на екран
public void displayForward() {
    System.out.print("Список: ");
    Link current = first;
    while (current != null)
 {
      current.displayLink();
      current = current.next;
    }
    System.out.println("");
  }

//Виведення елементів списку у зворотному порядку
  public void displayBackward() {
    System.out.print("Список зворотний: ");
    Link current = last;
    while (current != null){
      current.displayLink();
      current = current.prev;
    }
    System.out.println("");
  }

  public static void main(String[] args) {
    DoublyLinkedList theList = new DoublyLinkedList();
    Random ran = new Random();
    int n = ran.nextInt(20)+10;
    for(int i=0; i<n; i++)
    theList.insertLast(ran.nextInt(20)-10);
    theList.displayForward();
    System.out.println("Видаляємо перше входження 3");
    theList.deleteKey(3);
    theList.displayForward();
    System.out.println("Видаляємо усі від'ємні елементи");
    theList.delete();
    theList.displayForward();
    theList.displayBackward();

}}
```

```
Список: -9 4 9 -2 -5 4 1 3 9 -1 4 6 -8 -4 8 5 6 2
Видаляємо перше входження 3
Список: -9 4 9 -2 -5 4 1 9 -1 4 6 -8 -4 8 5 6 2
Видаляємо усі від'ємні елементи
Список: 4 9 4 1 9 4 6 8 5 6 2
Список зворотний: 2 6 5 8 6 4 9 1 4 9 4
```

■ ТЕД int. Надрукувати елементи дерева у зворотному порядку і окремо тільки елементи з крайніх листів дерева.

```
public class BinaryTreeTest {

  public static void main(String[] args) {
    new BinaryTreeTest().run();
  }

  static class Node {
    Node left;
    Node right;
    int value;
    public Node(int v) {
      value = v;
    }
  }

  public void run() {
   //Будуємо дерево
    Node root = new Node(5);
    System.out.println("Будуємо дерево зі значення в корені "
+ root.value);
    insert(root, 1);
    insert(root, 8);
    insert(root, 6);
    insert(root, 3);
    insert(root, 9);
    System.out.println("Елементи дерева у зворотному
порядку:");
    printInOrder(root);
    System.out.println("Елементи з крайніх листів дерева:");
    getLeaf(root);
  }

  public void getLeaf(Node node) {
      if (node.right==null)
      { System.out.printf("Знайдено лист - %d%n", node.value);
        return;
      }
      //інакше
      if (node.right!= null)
      //йдемо по правій гільці
      getLeaf(node.right);
```

```
      if (node.left != null)
      //йдемо по лівій гільці
       getLeaf(node.left);
    }

    public void insert(Node node, int value) {
    if (value < node.value) {
      if (node.left != null) {
        insert(node.left, value);
      } else {
        System.out.println("Додано " + value + " ліворуч від "
            + node.value);
        node.left = new Node(value);
      }
    } else if (value > node.value) {
      if (node.right != null) {
        insert(node.right, value);
      } else {
       System.out.println("Додано " + value + " праворуч від "
            + node.value);
        node.right = new Node(value);
      }
    }
  }

   public void printInOrder (Node node) {
    if (node != null) {
      printInOrder(node.right);
      System.out.println(node.value);
      printInOrder(node.left);
    }
  }
}
```

Будуємо дерево зі значення в корені 5
Додано 1 ліворуч від 5
Додано 8 праворуч від 5
Додано 6 ліворуч від 8
Додано 3 праворуч від 1
Додано 9 праворуч від 8
Елементи дерева у зворотному порядку:
9
8
6
5
3
1
Елементи з крайніх листів дерева:

Знайдено лист - 9
Знайдено лист - 6
Знайдено лист - 3

2.3 Контрольні питання

1. Що собою уявляє стек. Де така структура організації даних використовується?
2. Дайте характеристику АТД «черга». Які особливості створення черги на основі масиву?
3. Поясніть відмінності однозв'язного і двузв'язного списків. Яким чином вони задаються?
4. Що собою представляє структура бінарне дерево?
5. Яким чином визначається глибина бінарного дерева?

2.4 Варіанти завдань

1. ■ Використовуючи динамічну структуру «стек» для зберігання символів, визначте чи є задана фраза паліндромом, тобто такою, що однаково читається в обох напрямках (зліва направо та справа наліво).
   ■ ТЕД int. Надрукувати елементи з крайніх листів дерева у порядку їх зростання.
2. ■ Використовуючи динамічну структуру список, зберегти ряд чисел. Видалити елементи, які повторюються.
   ■ ТЕД int. Визначити число входжень деякого елемента у дерево.
3. ■ Сформувати динамічну структуру чергу, елементами якої є цифри. Виймаючи елементи з черги, надрукувати їх двійкові еквіваленти.
   ■ ТЕД double. Обчислити середнє арифметичне усіх елементів дерева.
4. ■ Побудувати динамічний список з елементів цілого типу за допомогою генератору випадкових чисел. Найти середньо арифметичне значення елементів списку.
   ■ ТЕД double. Замінити усі від'ємні елементи на їхні абсолютні значення.
5. ■ Використовуючи динамічну структуру список, зберегти ряд чисел. Видалити зі списку всі від'ємні елементи.
   ■ ТЕД char. Надрукувати елементи з усіх листів дерева.
6. ■ Задати два динамічних списки. Перевірити їх на рівність.
   ■ ТЕД char. Визначити максимальну глибину (тобто число гілок у самому довгому із шляхів від кореня до листів).
7. ■ Використовуючи динамічну структуру список, підрахувати кількість додатних і від'ємних чисел у списку.

■ ТЕД double. Знайти величину найбільшого елемента дерева.

8. ■ Використовуючи динамічну структуру «стек», зашифрувати зміст текстового файлу.

■ ТЕД char. Надрукувати елементи з крайніх листів дерева у порядку їх убування.

9. ■ Сформувати чергу з елементів цілого типу. Парні елементи піднести до квадрату. Роздрукувати вихідну і результуючі черги.

■ ТЕД char. Надрукувати список усіх елементів дерева і для кожного елемента – список номерів, де він зустрівся.

10. ■ Використовуючи динамічну структуру список, підрахувати суму від'ємних чисел у списку.

■ ТЕД char. Надрукувати всі елементи дерева у порядку їх убування.

11. ■ Використовуючи динамічну структуру список, підрахувати кількість цифр в заданому наборі символів.

■ ТЕД int. Визначити число додатних елементів у всіх правих гілках дерева.

12. ■ Використовуючи динамічну структуру список, перевірити чи є він упорядкованим набором чисел.

■ ТЕД int. Визначити число від'ємний елементів у всіх лівих гілках дерева.

13. ■ Використовуючи динамічну структуру список, підрахувати кількість руських малих літер в рядку.

■ ТЕД char. Надрукувати елементи у всіх вузлах дерева.

14. ■ Використовуючи динамічну структуру чергу, зашифрувати зміст тексту: кожний символ замінити його кодом + 1.

■ ТЕД int. Надрукувати тільки додатні елементи дерева.

15. ■ Використовуючи динамічну структуру чергу, перевести введену послідовність чисел в слово, яке складається з кодів Unicode.

■ ТЕД int. Надрукувати тільки від'ємні елементи дерева.

3 Прилади, устаткування та інструменти

Для виконання лабораторної роботи використовується ПЕОМ з установленим пакетом Sun Microsystems JDK 1.5 і вище та інтегрованим середовищем розробки BlueJ або Eclipse. Для написання програми на Java може бути використаний будь-який текстовий редактор, наприклад, Notepad, WordPad в MS Windows і ін.

4 Правила техніки безпеки та охорони праці

Правила техніки безпеки при виконанні лабораторної роботи регламентуються «Правилами техніки безпеки при роботі в комп'ютерній лабораторії».

5 Порядок проведення лабораторної роботи

Для виконання роботи кожен студент повинен:

1. Відповісти на контрольні питання та пройти усне опитування за теоретичним матеріалом лабораторної роботи;
2. Пройти інструктаж за правилами охорони праці;
3. Запустити на комп'ютері інтегроване середовище розробки BlueJ;
4. Отримати варіант завдання у викладача;
5. Скласти алгоритм розв’язання задачі;
6. Записати код програми на комп’ютері;
7. Відкомпілювати програму та виправити всі помилки;
8. Запустити програму на виконання;
9. Отримати результати роботи програми і показати їх викладачу;
10. Підготувати і захистити звіт до лабораторної роботи.

6 Оформлення і захист звіту

Підготовлений до захисту звіт до лабораторної роботи повинен містити:

1. титульний лист, де вказані номер і назва лабораторної роботи, відомості про виконавця;
2. номер варіанта роботи та текст завдання;
3. відповіді на контрольні запитання до лабораторної роботи;
4. текст програми алгоритмічною мовою Java;
5. лістинг результатів виконання програми.

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни “Алгоритмізація і програмування”
частина II
для студентів I року денної форми навчання
Спеціальність – комп’ютерні науки.

Укладачі: доц. Кузніченко С.Д., доц. Коваленко Л.Б.

Підп. до друку Формат 60x84/16 Папір офс.

Умовн. а.а. Тираж Замовл.

Одеський державний екологічний університет,
65016, м. Одеса, вул. Львівська, 15