ЗАТВЕРДЖУЮ
Завідувач кафедри управління проектами, інформаційних технологій та телекомунікацій, д.т.н., професор
Є. В. Мартин
«_ _»_____20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення практичного заняття з студентами (курсантами) 1-го курсу спеціальності 122 «Комп'ютерні науки» з дисципліни «Основи програмування та алгоритмізація»

Практичне заняття 4.12: «Реалізація алгоритмів лінійного сортування»

Мета: здобути навики щодо реалізації алгоритмів лінійного сортування.

Час: 2 годин

Місце заняття: комп'ютерна лабораторія

Матеріальне забезпечення:
- ПК з відповідним ПЗ та доступом до мережі Internet;
- Методичні рекомендації щодо виконання практичної роботи.

Після практичного заняття студенти (курсанти) повинні:
вміти: реалізовувати алгоритми лінійного сортування.
знати: особливості застосування алгоритмів лінійного сортування.

Література:
1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з : http://virt.ldubgd.edu.ua/course/view.php?id=1672

План заняття:
1. Складання тестових завдань за темою попередньої лекції.
2. Реалізація алгоритмів лінійного сортування мовою програмування Java.

1. Складання тестових завдань за темою попередньої лекції.

Для складання тестових завдань необхідно перейти у відповідний розділ дисципліни «Основи програмування та алгоритмізація» у Віртуальному навчальному середовищі (ННІ цивільного захисту/спеціальність «Комп'ютерні науки»/бакалавр/І курс/ІІ семестр/Основи програмування та алгоритмізація/Тема 4. Алгоритми/Контрольна частина/Тестове завдання за темою 4.11.)

Слід звернути увагу, що час та період складання тестового завдання обмежено.

2. Реалізація алгоритмів лінійного сортування мовою програмування Java.

Перед виконання практичного завдання слід повторити лекцію «Лінійне сортування», та нагадати особливості роботи процедури CountingSotr і RadixSort. Розглянемо код алгоритму сортування підрахунком з детальним аналізом усіх методів:

```
import java.util.Arrays;

public class CountingSort {

	static void sort(int arr[]) {
		int n = arr.length; // записуємо у змінну n кількість елементів масиву
		int output[] = new int[n]; // створюємо вихідний масив на n елементів

		// створюємо додатковий масив для підрахунку кількості повторень
		// елементів та наповнюємо його "0"
		int count[] = new int[11];
		for (int i = 0; i < count.length; ++i)
			count[i] = 0;

		// проводимо підрахунок повторюваності елементів у масиві що подано на
		// сортування
		for (int i = 0; i < n; ++i)
			++count[arr[i]];

		// вивід у консоль значень кількості повторюваності елементів у масиві
		// для перевірки працездатності коду
		System.out.println("Значення повторюавності елементів у масиві: ");
		for (int i = 0; i < n; ++i)
			System.out.print(count[i] + " ");
		System.out.println();

		// сумуємо значення комірок count[i] та count[i-1]
		for (int i = 1; i <= 10; ++i)
			count[i] += count[i - 1];

		// вивід у консоль значень суми комірок [i] та [i-1] для перевірки
		// працездатності коду
		System.out.println("Значення суми count[i] та count[i-1]: ");
		for (int i = 0; i < n; ++i)
			System.out.print(count[i] + " ");
		System.out.println();

		// побудова вихідного масиву із одночасним зменшенням значення комірки
		// [i] масиву count на 1
		for (int i = 0; i < n; ++i) {
			output[count[arr[i]] - 1] = arr[i];
			--count[arr[i]];
		}
		for (int i = 0; i < n; ++i)
			arr[i] = output[i];
	}
```

```
	// головний метод: створює масив на 10 елементів, наповнює його випадковими
	// значеннями, викликає метод сортування та виводить у консоль вхідний і
	// вихідний варіанти масиву
	public static void main(String args[]) {
		int[] arr = new int[10];
		for (int i = 0; i < arr.length; i++) {
			arr[i] = ((int) (Math.random() * 11));
		}
		System.out.println("Вхідний масив: ");
		System.out.println(Arrays.toString(arr));
		sort(arr);
		System.out.println("Вихідний масив: ");
		System.out.println(Arrays.toString(arr));
	}
}
```

На деяких інтернет ресурсах зустрічаються інші варіанти реалізації алгоритму сортування підрахунком. Розглянемо один з найбільш розповсюджених:

```
import java.util.Arrays;
public class CountingSort {
	public static void sort(int[] arr) {
		int N = arr.length;
		if (N == 0)
			return;
		int max = arr[0], min = arr[0];
		for (int i = 1; i < N; i++){
			if (arr[i] > max)
				max = arr[i];
			if (arr[i] < min)
				min = arr[i];
		}
		int range = max - min + 1;
		if (range > 10) {
			System.out.println("\nError : Range too large for sort");
			return;
		}
		int[] count = new int[range];
		for (int i = 0; i < N; i++)
			count[arr[i] - min]++;
		for (int i = 1; i < range; i++)
			count[i] += count[i - 1];
		int j = 0;
		for (int i = 0; i < range; i++)
			while (j < count[i])
				arr[j++] = i + min;
	}
	public static void main(String[] args) {
		int[] arr = new int[10];
		for (int i = 0; i < arr.length; i++) {
			arr[i] = ((int) (Math.random() * 11));
		}
		System.out.println(Arrays.toString(arr));
		sort(arr);
		System.out.println(Arrays.toString(arr));
	}
}
```

Для закріплення пройденого матеріалу в якості практичного завдання необхідно реалізувати алгоритм сортування за розрядами використовуючи в основі один із запропонований алгоритмів CountingSort. Псевдокод процедури RadixSort:

```
RadixSort(A, d)
1  for i ← 1 to d
2      do Стійке сортування масиву A за i-м розрядом
```

Отже **умова завдання**: створити масив на 10 елементів, наповнити його випадковими значеннями від 0 до 888. Відсортувати масив за допомогою процедури RadixSort використовуючи один із розглянутих алгоритмів сортування підрахунком.

Код написаної програми необхідно завантажити у віртуальне навчальне середовище в категорію «Реалізація алгоритмів лінійного сортування». Слід зауважити, що період виконання завдання обмежений.

Методичну розробку розробив:
Заступник начальника кафедри УПІТтаТ
майор служби цивільного захисту О.В. Придатко

Методична розробка обговорена на засіданні кафедри УПІТтаТ
Протокол № ____ від "___" _____ 20__ р.