# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №4**

**«Алгоритм прямого пошуку стрічки»**

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

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняла:

доц. Кортєєва Т.О.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Алгоритм прямого пошуку стрічки.

**Мета роботи**: Вивчити алгоритм прямого пошуку стрічки. Здійснити програмну реалізацію даного алгоритму.

## 1. Теоретичні відомості

Нехай задано масив *S* з *n* елементів та масив *P* з *m* елементів, причьому *m<=n.* Необхідно знайти перше входження масива *P* у масив *S* . Алгоритм зводиться до повтору порівнянь окремих елементів.

Алгоритм R

R1. Встановити *і=1.. n-m,j=1..m*.

R2. Якщо *S[i] = P[j]* , то зафіксувати перше співпадіння *k=i*, та перевірити спів падіння всього масиву *P* у масиві *S.* При першому неспівпадінні відмінити значення *к* та продовжити пошук.

R3. Кінець.

Кількість порівнянь дорівнює ***n\*m*.**

**2. Текст програми на мові програмування JAVA**

package \_asd\_lab4;

import java.util.\*;

/\*\*

\* Л/р № 4 Алгоритм прямого пошуку стрічки

\* @author Taras

\*/

public class \_ASD\_Lab4 {

//чи показувати проміжні результати

static final boolean showIntermediateResults = true;

//вивід покрокової інформації

static void message(Object obj) {

if(showIntermediateResults) System.out.println(obj);

}

//пошук першого входження підстроки word в строці text

//повертає позицію першого символа, що співпав, якщо

//підстроку не знайдено - повертає -1

static int substringSearch(char[] text, char[] word) {

//якщо пусте слово та пустий текст - то позиція входження = 0

//якщо пусте слово в не пустому тексті –

//то позиція входження = -1

if(word.length == 0) return text.length == 0 ? 0 : -1;

for(int i = 0; i < text.length; i++) {

int n = 0;

message("\ni = " + i);

for(int j = 0; (j < word.length)&&(i+j < text.length); j++) {

message("" + word[j] + " == " + text[i+j] + " --> "

+ (word[j] == text[i+j]));

//якщо не співпав наступний символ - не продовжуємо

if(word[j] != text[i+j]) break;

n++;

}

//якщо всі символи слова співпали

if(n == word.length) {

message("res = " + i);

return i;

}

}

return -1; // не знайдено

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter text: ");

String text = sc.nextLine();

System.out.println("Enter word: ");

String word = sc.nextLine();

int pos = substringSearch(text.toCharArray(), word.toCharArray());

if(pos < 0)

System.out.println("\nWord is not found in the text.");

else

System.out.println("\nEntry position: " + pos);

}

}

**3. Результат виконання програми**

Запустимо програму на виконання без виведення проміжних результатів на екран (рис. 3.1 - 3.2):

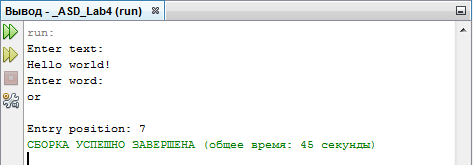


Рис. 3.1. Результат виконання програми

(без проміжних результатів)

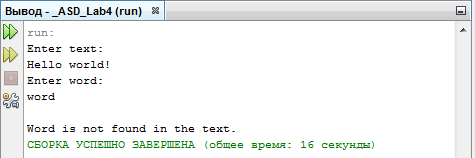


Рис. 3.2. Результат виконання програми

(без проміжних результатів)

Запустимо програму на виконання з виведенням проміжних результатів на екран (рис. 3.3 - 3.4):

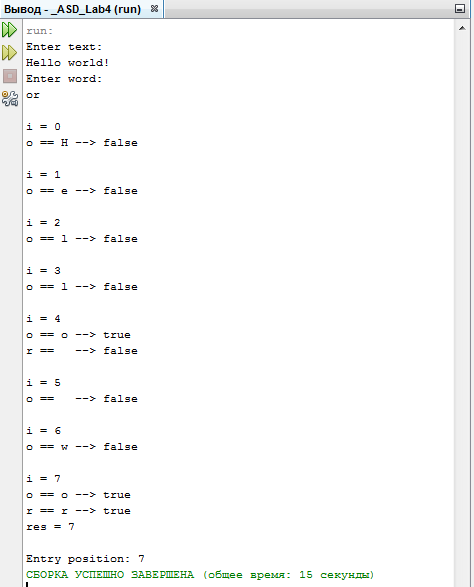


Рис. 3.3. Результат виконання програми

(з проміжними результатами)

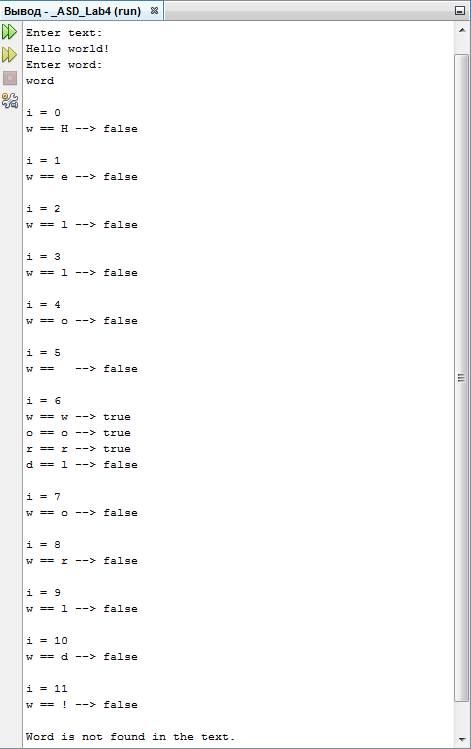


Рис. 3.4. Результат виконання програми

(з проміжними результатами)

## ВИСНОВКИ

На даній лабораторній роботі я навчився програмно реалізовувати алгоритм прямого пошуку стрічки, удосконалив свої навички у написанні підпрограм для роботи з стрічковими масивами у JAVA.