Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

Кафедра комп'ютерних систем і мереж

Підлягає поверненню на кафедру

ЗАСОБИ ШТУЧНОГО ІНТЕЛЕКТУ В СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

Методичні вказівки до лабораторних робіт

Укладач Баловсяк С. В.

Чернівці
Чернівецький національний університет
2014

УДК 004.8 (076.5)
ББК 32.815я73
З - 363

Друкується за ухвалою редакційно-видавничої ради
Чернівецького національного університету
імені Юрія Федьковича

З - 363 **Засоби** штучного інтелекту в спеціалізованих комп'ютерних системах : методичні вказівки до лабораторних робіт / укл. Баловсяк С. В. – Чернівці : Чернівецький національний університет, 2014. – 75 с.

Дисципліна „Засоби штучного інтелекту в спеціалізованих комп'ютерних системах" читається студентам спеціальності 8.05010201 "Комп'ютерні системи та мережі". Методичні вказівки сприятимуть підготовці до виконання п'яти лабораторних робіт, які дозволять практично засвоїти теоретичний матеріал дисципліни. Лабораторні роботи адаптовані до кредитно-модульної системи навчання.

Для студентів вищих навчальних закладів.

УДК 004.8 (076.5)
ББК 32. 815я73

ВСТУП

Завдання навчальної дисципліни „ Засоби штучного інтелекту в спеціалізованих комп'ютерних системах" – на основі отриманих теоретичних знань виробити у студентів уміння користуватися існуючими, а також створювати власні системи розпізнавання образів, штучні нейронні мережі, експертні системи, системи нечіткої логіки та генетичні алгоритми.

Метою навчальної дисципліни є вивчення студентами методів і способів створення засобів штучного інтелекту в спеціалізованих комп'ютерних системах, отримання відомостей про основи штучного інтелекту, методи представлення знань і баз знань, системи нечіткої логіки, будову та можливості використання експертних систем, основні поняття про системи розпізнавання образів, штучні нейронні мережі та генетичні алгоритми.

У результаті вивчення дисципліни студенти повинні:

▪ ЗНАТИ основні методи розпізнавання образів, принципи нечіткого логічного виведення, основні методи представлення знань, будову експертних систем, будову і принципи функціонування штучних нейронних мереж, основи генетичних алгоритмів.

▪ ВМІТИ проектувати і створювати бази знань та експертні системи, використовувати нечітке логічне виведення; створювати, навчати і використовувати штучні нейронні мережі, розв'язувати оптимізаційні задачі за допомогою генетичних алгоритмів; здійснювати розпізнавання образів, зокрема зображень.

Кожна лабораторна робота містить: тему і мету роботи; опис обладнання та програмних засобів, потрібних для виконання даної лабораторної роботи; список літератури; короткі теоретичні відомості, необхідні для виконання лабораторної роботи; порядок виконання роботи; контрольні запитання для самоперевірки.

Звіт по виконаній лабораторній роботі повинен містити:

– номер лабораторної роботи;

– тему лабораторної роботи;

– мету роботи;

– інструментальні засоби, потрібні для виконання лабораторної роботи;

– порядок виконання лабораторної роботи з коротким аналізом виконаних етапів роботи;

– результати виконання роботи у вигляді тексту, схем, графіків, таблиць, схем алгоритмів.

Дані вказівки до п’яти лабораторних робіт призначені для закріплення практичних навичок, набутих при вивченні дисципліни „Засоби штучного інтелекту в спеціалізованих комп’ютерних системах”.

Мета роботи № 1 „Розпізнавання зображень символів, отриманих з USB-відеокамери” полягає у вивченні принципів розпізнавання образів, зокрема зображень символів, за допомогою методу зондів. У роботі використовується програма, яка зчитує зображення символів з відеокамери.

В роботі № 2 „Комп’ютерна реалізація фатичного діалогу” виконується реалізація фатичного діалогу у вигляді програми. Використовується п’ять варіантів порівнянь повідомлень користувача зі зразками речень.

У роботі № 3 „Навчання штучної нейронної мережі методом зворотного розповсюдження помилки” розглядаються принципи побудови та навчання штучних нейронних мереж методом зворотного розповсюдження помилки; виконується розпізнавання зображень символів за допомогою нейромережі.

Мета лабораторної роботи № 4 „Використання генетичних алгоритмів” – вивчення принципів розв’язання оптимізаційних задач за допомогою генетичних алгоритмів, набуття навичок програмування генетичних алгоритмів при моделюванні еволюції форми дерева.

Робота № 5 „Проектування експертної системи” передбачає вироблення навичок проектування експертної системи, зокрема визначення складу знань та засвоєнні технології здобуття експертних знань, які стосуються визначеної предметної області.

Вищевказані лабораторні роботи входять до складу трьох змістових модулів: змістовий модуль 1 містить роботи № 1 та № 2, змістовий модуль 2 – роботи № 3, № 4, змістовий модуль 3 – роботу № 5. Рекомендується оцінювати знання та вміння студентів за результатами виконання лабораторних робіт так: робота № 1 – максимальна оцінка 8 балів, № 2 – 8 балів, № 3 – 8 балів, № 4 – 8 балів, № 5 – 8 балів.

ЛАБОРАТОРНА РОБОТА № 1

РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ СИМВОЛІВ, ОТРИМАНИХ З USB-ВІДЕОКАМЕРИ

Мета: вивчити принципи розпізнавання образів, зокрема зображень символів, за допомогою методу зондів.

Завдання: згідно варіанту отримати зображення символів за допомогою *USB*-відеокамери, створити набір зондів для розпізнавання вказаних класів символів, написати додаткові процедури для програми *„USB_Video_Rec_11”* і виконати розпізнавання символів за їх зображеннями методом зондів.

Обладнання: відеокамера *„G-Cube GWL-835B”*, штатив відеокамери, персональний комп’ютер.

Програмне забезпечення: програма *„USB_Video_Rec_11”*, середовище розробки програм *Delphi*.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Задачі розпізнавання образів

Розпізнавання образів є однією з найфундаментальніших проблем теорії інтелектуальних систем і має величезне практичне значення. Розглянемо основні поняття, які використовуються в задачах розпізнавання образів [1 - 4].

Образи – це об’єкти і явища навколишнього середовища або продукти розумової діяльності людини. Прикладом образів можуть бути: люди (їх зображення, голоси, почерки, пульс); літери; літаки, автомобілі, рослини, тварини; суспільні, економічні явища і процеси.

Ознаки образу – це якісні та кількісні параметри, сукупність яких дозволяє відрізнити один образ від іншого. Ознаки можуть бути суттєвими й несуттєвими, кількісними, якісними, апріорними, апостеріорними. Відомі ознаки до досліду – апріорні, а в результаті досліду отримуються апостеріорні або емпіричні ознаки.

Початковий словник ознак – перелік ознак образу, які отримуються за допомогою сенсорів.

Однотипні образи – група образів зі скінченною кількістю спільних ознак, об’єднаних приблизно однаковими функціями (літаки, квіти, букви, цифри).

Клас образів – це підмножина образів із найбільш схожими суттєвими ознаками.

Приклади класів образів: однотипні образи „Букви українського алфавіту” розділяються на класи символів „А”, „Б”, „В” і т.д. Часто до m існуючих класів додається (m +1)-й „нерозпізнаний образ”. Це означає певну невизначеність, що іноді краще, ніж помилкове визначення класу образу.

Розпізнавання образів **(РО)** – процес поділу образів на класи.

Мета РО – розробка системи РО, яка б наближувалася за своїми можливостями до можливостей людини.

1.2. Математична постановка задачі розпізнавання

Нехай дана множина об’єктів T_i, де i = 1,... q (кількість об’єктів q може бути практично нескінченною); об’єкти, які підлягають розпізнаванню, називають також зразками (*Pattern*).

Усі об’єкти поділені на класи (підмножини) C_k, де k = 1,... m. У багатьох методах розпізнавання кожному класу відповідає один або кілька еталонів (*Etalon*). Об’єкти задаються значеннями ознак X_j, де j = 1,... n; $T_i = \{X_1(T_i),... X_n(T_i)\}$.

Задача розпізнавання полягає у знаходженні ступеня належності об’єкта T_i до кожного з класів C_k. Об’єкт T_i вважається належним до того класу C_k, для якого коефіцієнт подібності (коефіцієнт кореляції) $K_C(T_i, C_k)$ максимальний. Значення коефіцієнта подібності $K_C(T_i, C_k)$ залежить від відстані $D(T_i, C_k)$ у n-вимірному просторі ознак X між відповідним об’єктом і класом:

$$K_C(T_i, C_k) = n - D(T_i, C_k), \qquad (1.1)$$

де в нашому випадку відстань $D(T_i, C_k) = \sum_{j=1}^{n} \left| X_j(T_i) - X_j(C_k) \right|$.

Розглянемо, наприклад, множину об’єктів T „Транспортні засоби”:

- ознаки X_j: кількість коліс, наявність двигуна, вага вантажу;
- класи C_k: велосипед, мотоцикл, легковий автомобіль, вантажний автомобіль.

Апріорна інформація в задачі розпізнавання часто задається у вигляді таблиці (табл. 1.1).

Наведемо приклади розпізнавання об’єктів з ознаками $T_1(3, 1, 150)$ /мотоцикл/ та $T_2(4, 0, 50)$ /велосипед/ (табл. 1.2). У даному випадку значення коефіцієнта кореляції $K_C(T_i, C_k)$ визначається як кількість спільних ознак відповідного об’єкта і класу.

Типові задачі розпізнавання:

1. Технічна діагностика.
2. Медична діагностика.
3. Розпізнавання літер (друкованого й рукописного текстів).

4. Розпізнавання мови.
5. Робототехніка (технічний зір).
6. Охоронні системи (розпізнавання облич, відбитків пальців).

Таблиця 1.1

Значення ознак для множини об'єктів T „Транспортні засоби"

Класи	Назви класів	Ознаки		
		X_1	X_2	X_3
		Кількість коліс	Наявність двигуна (0/1)	Вага вантажу, кг
C_1	Велосипед	2	0	<100
C_2	Мотоцикл	2, 3	1	<200
C_3	Легковий автомобіль	4	1	(>200) AND (<500)
C_4	Вантажний автомобіль	4, 6, 8, 10	1	>1000

Таблиця 1.2

Значення коефіцієнтів подібності K_C (T_i, C_k)

Класи C_k	Назви класів	Об'єкти T_i	
		T_1 (3, 1, 150)	T_2 (4, 0, 50)
C_1	Велосипед	0	2
C_2	Мотоцикл	3	1
C_3	Легковий автомобіль	2	1
C_4	Вантажний автомобіль	1	1

1.3. Метод зондів

У методі зондів виконується розпізнавання у просторі ознак. Даний метод використовується для розпізнавання цифр та літер, навіть написаних від руки та з деякими відхиленнями у розмірах і стилі написання. Уперше метод зондів запропонований британським вченим Д. Даймондом у 1958 р. і використовувався для обробки чеків на сплату міжнародних телефонних розмов [2].

Розглянемо принцип використання зондів у припущенні, що наша система РО повинна розпізнавати образи лише чотирьох великих друкованих літер „П О Р Т" (рис. 1.1).

Зображення літер можуть записуватися в комп'ютер з відеокамери, сканера або графічних файлів. Після бінаризації зображення записуються в пам'ять комп'ютера як прямокутні матриці чисел („0" або „1"), де „0" означає належність відповідного пікселя до символу, а „1" – до фону. Можна вважати, що кожний зонд 1, 2, 3 (рис. 1.1) є зв'язною областю пікселів певної форми (наприклад, прямокутника).

Якщо хоча б один з пікселів зонда належить символу, то зонд вважається активним (код „1"), а інакше – пасивним (код „0"). Кожному зображенню одного класу (одній літері) звичайно відповідає одна комбінація активних і пасивних зондів. Таким чином, зонди видають двійковий код, через який визначається клас символів (табл. 1.3).

У дійсності для визначення літер всього алфавіту зонди мають складніший вигляд у порівнянні з показаними на рис. 1.1. Пояснюється це тим, що розглядається більша кількість літер і одночасно враховується можливе відхилення форми літери від еталона. Перед накладанням зондів на зображення символу також відбувається їх центрування.

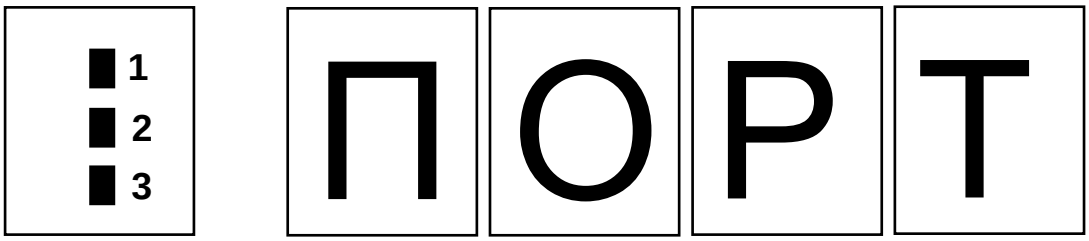


Рис. 1.1. Використання зондів для розпізнавання літер „ПОРТ"

Таблиця 1.3

Коди зондів для літер „ПОРТ"

Літери	Зонди		
	1	**2**	**3**
П	1	0	0
О	1	0	1
Р	1	1	0
Т	1	1	1

У вдосконалених варіантах методу зондів враховуються можливі деформації зображень: стиснення або розтяг за висотою і шириною, локальні спотворення контуру об'єкта у визначених межах; нахил або поворот об'єкта на деякий кут [5].

1.4. Програма „USB_Video_Rec_11”

Програма „*USB_Video_Rec_11*” створена в середовищі *Delphi* і призначена для зчитування та обробки зображень з *USB*-відеокамери (рис. 1.2). Програма взаємодіє з відеокамерою за допомогою драйверів, які повинні бути попередньо встановлені на ПК [6].

У роботі використовується відеокамера „*G-Cube GWL-835B*”, яка забезпечує максимальну роздільну здатність 640×480 пікселів, зчитування 30 кадрів за секунду, фокусування від 10 см до безкінечності. Окремі кадри (*Frame*) відеопотоку зчитуються з інтервалом, що встановлюється компонентом „*Timer1_Video”* (за замовчуванням 112 мс). Дозвіл за зчитування кадрів встановлюється командою „*Старт*”, а забороняється – командою „*Стоп*”. Зчитані або „захоплені” (*capture*) з відеопотоку кадри відображаються на зображенні „*Image_Video*”. Програма використовує роздільну здатність зображень 320×240 пікселів при максимальній роздільній здатності відеокамери 640×480 пікселів.

Захоплення відео у програмі виконується за допомогою функцій *Windows API,* які містять префікс *cap* (*capture*). Основна функція, яка використовується при захопленні відео, це „*capCreateCaptureWindowA*”. Вона звертається до стандартного драйвера „*AVICAP32.DLL*” і створює графічне вікно, призначене для показу відеопотоку. Цьому вікну посилаються всі керуючі повідомлення (*SendMessage*). Роздільну здатність та інші параметри відео можна встановити командою „*Video_Parameters*” на формі „*Опції*” (рис. 1.2).

Окремі зображення з відеокамери (рис. 1.3) записуються у вхідне зображення „*Image_Input*” командою „*Зчитати*”. Такі зображення можуть записуватися у файл командою „*Зберегти*”. Зображення „*Image_Input*” також зчитується з файла командою „*Відкрити*” (без використання відеокамери). На зображенні „*Image_Input*” (розміром *qXi* × *qYi* пікселів) користувач може вибрати певний об’єкт (наприклад, символ) за допомогою *маркера* (хрестик). Виділення квадратного фрагмента зображення «*Image_Fragment*» (центр вказується маркером) із розміром „*Fragment_Size*” виконується командою „*Фрагмент*”. Далі здійснюється сегментація зображення фрагмента (розміром *qFx* × *qFx* пікселів), у результаті чого отримується виділяється символ на зображенні „*Image_Symbol*”.

Розпізнавання виділеного об’єкта виконується командою „*Розпізнати*” (процедура „*p_Rec_Image*”) і містить такі кроки:

1. Виділення на зображенні „*Image_Symbol*” прямокутника символу розміром *qSx* × *qSy* пікселів.

2. Зчитування координат зондів та зображення зондів „*Image_Z*” (*qZx*×*qZy* пікселів) (рис. 1.4).
3. Поворот прямокутника символу на вказаний кут та його масштабування до розмірів зображення зондів, унаслідок чого отримується зображення „*Image_Symbol*_1”.

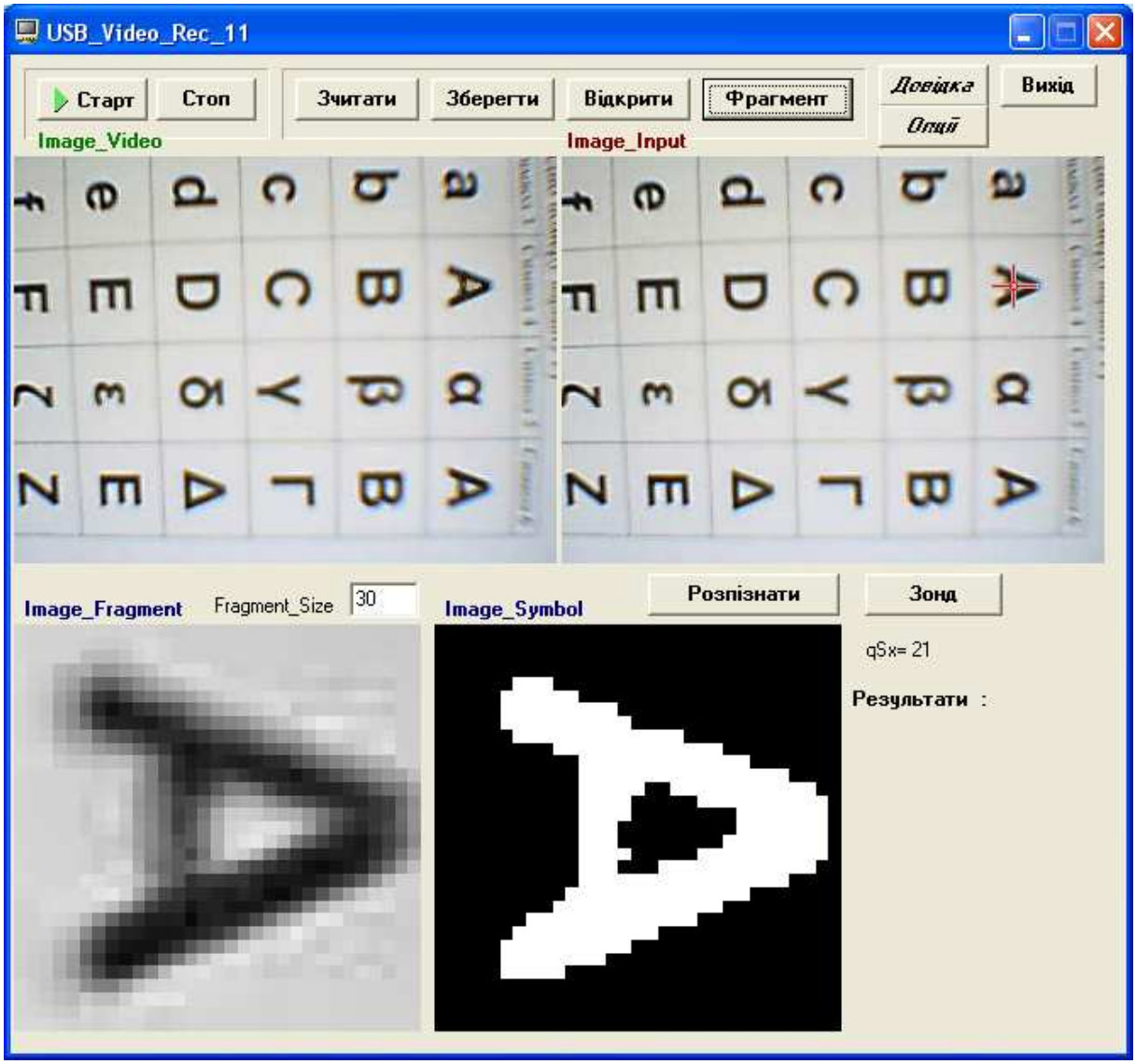


Рис. 1.2. Головна форма програми „*USB_Video_ Rec_11*”

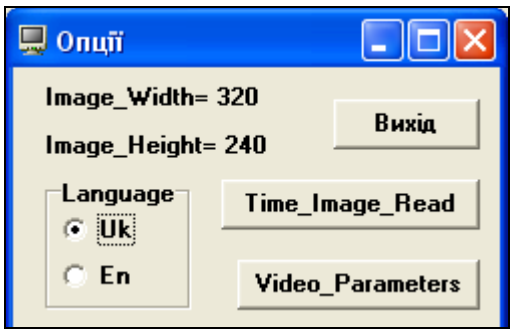
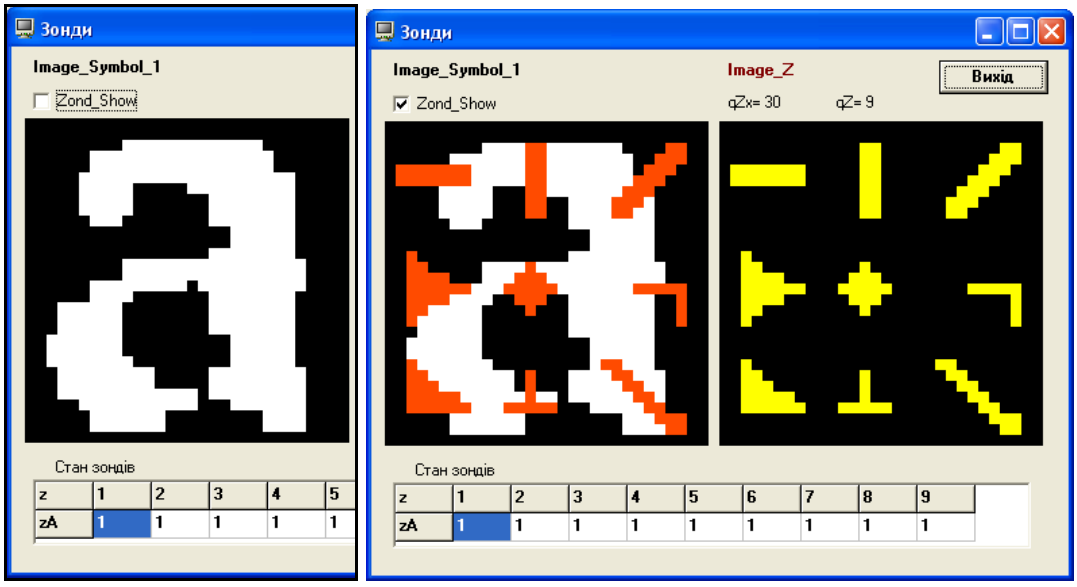
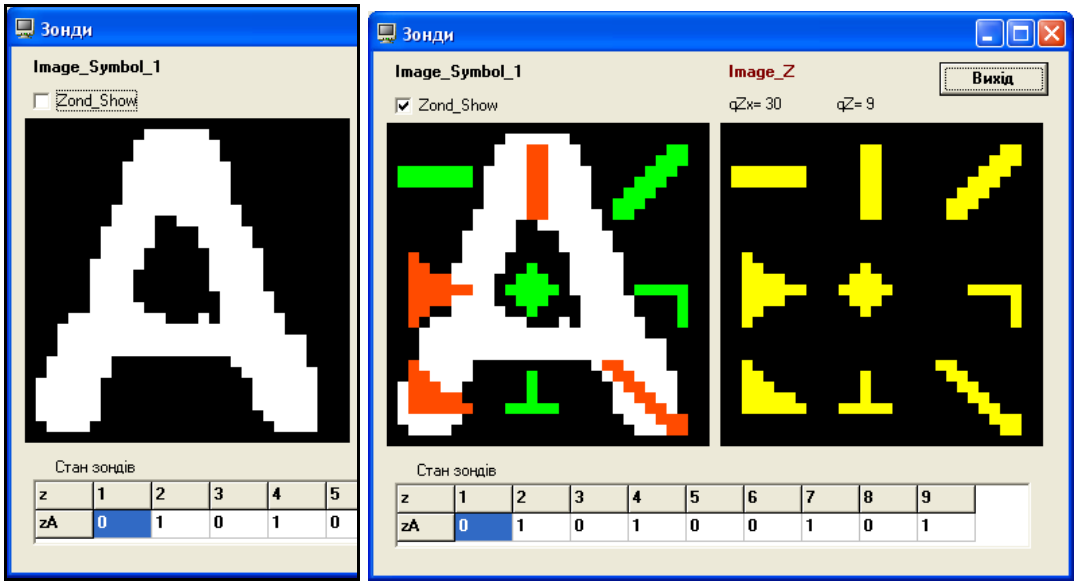


Рис. 1.3. Форма „*Опції*”

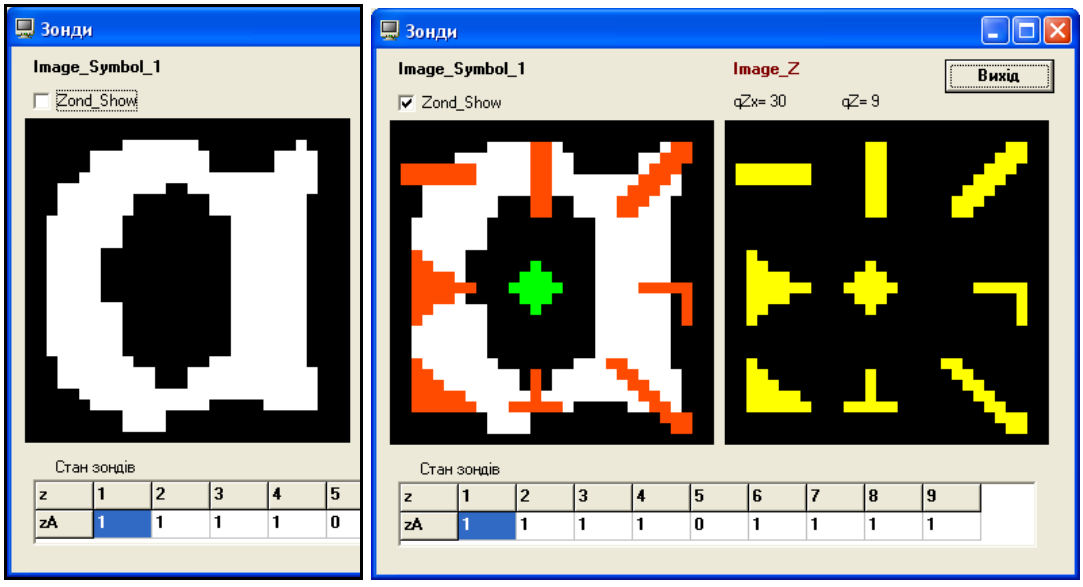
Рис. 1.4. Відеокамера „*G-Cube GWL-835B*”



а



б



в

Рис.1.4. Стани зондів для зображень символів „a”, „A”, „α”

4. Накладання зондів на зображення „*Image_Symbol*_1”: якщо зонд перекриває хоча б один піксель символу, то він набуває значення „1”, а інакше – „0” (рис. 1.4).
5. Порівняння отриманого стану зондів, записаного в масив $A[z]$ (номер зонду $z = 1 .. qZ$), з попередньо визначеними станами зондів $E[z,k]$ для певних класів символів C_k. Розпізнаний символ T_i належить до того класу символів C_k, для якого отримано максимальне значення коефіцієнта подібності $K_C(T_i, C_k)$ (на основі формули 1.1).

$$K_C(T_i, C_k) = qZ - D(T_i, C_k), \qquad (1.2)$$

де qZ = n – кількість ознак символу (кількість зондів);

відстань $D(T_i, C_k) = \sum_{z=1}^{qZ} \left| A_z(T_i) - E_z(C_k) \right|$.

Примітка. Крок № 5 у програмі „*USB_Video_Rec_11*” відсутній, тому його потрібно реалізувати програмно в процедурі „*p_Rec_Image*”.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів V 1...V 30

1. Ознайомитися з теоретичними відомостями.
2. Під’єднати відеокамеру до *USB* порту комп’ютера, за допомогою штатива встановити її на висоті приблизно 17 см над поверхнею стола. Під відеокамерою на столі розмістити аркуш паперу із зображеннями символів. За допомогою програми „*USB_Video_Rec_11*” зчитати з відеокамери зображення шести символів згідно з варіантом *V* (один рядок *V* з табл. 2.1). При відсутності відеокамери зчитувати зображення з файлів (папка „*Image*”).
3. Отримані шість зображень символів (еталони) записати у файли й масштабувати за допомогою графічного редактора до розміру $qZx \times qZy$ пікселів ($qZx = qZy = 30$). Для кожного класу символів, а саме для зображень відповідних еталонів вибрати таке розміщення qZ зондів на символі й такі їхні розміри, які б дозволяли найкраще відрізняти символи різних класів між собою (для непарних варіантів qZ = 9, для парних qZ = 10). Як перше наближення можна вибрати рівномірне розміщення зондів, яке потім потрібно коректувати.
4. Створене зображення із qZ зондів зберегти у файлі формату *.bmp. Зображення зондів виконати червоним кольором на чорному фоні (рис. 1.4). Після цього файл зондів формату *.bmp потрібно записати у текстовий файл як матрицю яскравостей пікселів за допомогою програми „*Image_to_File*” (рис. 2.1).

Таблиця 2.1

Символи (згідно з номером варіанта *V*)

V	Символ 1	Символ 2	Символ 3	Символ 4	Символ 5	Символ 6
0	а	А	a	A	α	A
1	б	Б	b	B	β	B
2	г	Г	c	C	γ	Г
3	д	Д	d	D	δ	Δ
4	е	Е	e	E	ε	E
5	є	Є	f	F	ζ	Z
6	ж	Ж	g	G	η	H
7	з	З	h	H	θ	Θ
8	и	И	i	I	ι	I
9	і	І	j	J	к	K
10	ї	Ї	k	K	λ	Λ

Продовження таблиці 2.1

Символи (згідно з номером варіанта *V*)

V	Символ 1	Символ 2	Символ 3	Символ 4	Символ 5	Символ 6
11	й	Й	l	L	μ	M
12	к	К	m	M	ν	N
13	л	Л	n	N	ξ	Ξ
14	м	М	o	O	ο	O
15	н	Н	p	P	π	П
16	о	О	q	Q	ρ	P
17	п	П	r	R	ς	Σ
18	р	Р	s	S	σ	Σ
19	с	С	t	T	τ	T
20	т	Т	u	U	υ	Y

Продовження таблиці 2.1

Символи (згідно з номером варіанта *V*)

V	Символ 1	Символ 2	Символ 3	Символ 4	Символ 5	Символ 6
21	у	У	v	V	φ	Φ
22	ф	Ф	w	W	χ	Χ
23	х	Х	x	X	ψ	Ψ
24	ц	Ц	y	Y	ω	Ω
25	ч	Ч	z	Z	β	В
26	ш	Ш	1	6	γ	Г
27	щ	Щ	2	7	δ	Δ
28	ю	Ю	3	8	ε	Е
29	я	Я	4	9	ζ	Z
30	ь	Ь	5	0	η	Н

В одержаній матриці чисел потрібно замінити яскравості пікселів зондів на значення їх номера (нумерацію зондів виконувати зліва направо і зверху вниз). На початку текстових файлів зондів потрібно записати розміри зображення зондів (*qZx* × *qZy* пікселів). Для різних варіантів *V* вибрати нижченаведені форми зондів (табл. 2.2).

Таблиця 2.2

Форма зондів для варіантів *V*

V	Форма зондів
1	Горизонтальні лінії товщиною в 1 піксель
2	Вертикальні лінії товщиною в 1 піксель
3	Лінії під кутом в 45º і товщиною в 1 піксель
4	Прямокутні трикутники
5	Рівносторонні трикутники
6	Ромби
7	Г-подібні зонди з товщиною лінії в 1 піксель
8	Т-подібні зонди з товщиною лінії в 1 піксель
9	Горизонтальні лінії товщиною в 2 пікселі
10	Вертикальні лінії товщиною в 2 пікселі
11	Лінії під кутом в 45º і товщиною в 2 пікселі
12	Г-подібні зонди з товщиною лінії в 2 пікселі
13	Т-подібні зонди з товщиною лінії в 2 пікселі
14	Горизонтальні лінії товщиною в 1 піксель і квадрати
15	Вертикальні лінії товщиною в 1 піксель і квадрати
16	Лінії під кутом в 45º і товщиною в 1 піксель і квадрати
17	Горизонтальні лінії товщиною в 1 піксель і трикутники
18	Вертикальні лінії товщиною в 1 піксель і трикутники
19	Лінії під кутом в 45º і товщиною в 1 піксель і трикутники
20	Горизонтальні лінії товщиною в 1 піксель і ромби
21	Вертикальні лінії товщиною в 1 піксель і ромби
22	Лінії під кутом в 45º і товщиною в 1 піксель і ромби
23	Горизонтальні лінії товщиною в 2 пікселі і квадрати
24	Вертикальні лінії товщиною в 2 пікселі і квадрати
25	Лінії під кутом в 45º і товщиною в 2 пікселі і квадрати
26	Горизонтальні лінії товщиною в 2 пікселі і трикутники
27	Вертикальні лінії товщиною в 2 пікселі і трикутники
28	Лінії під кутом в 45º і товщиною в 2 пікселі і трикутники
29	Горизонтальні лінії товщиною в 2 пікселі і ромби
30	Вертикальні лінії товщиною в 2 пікселі і ромби

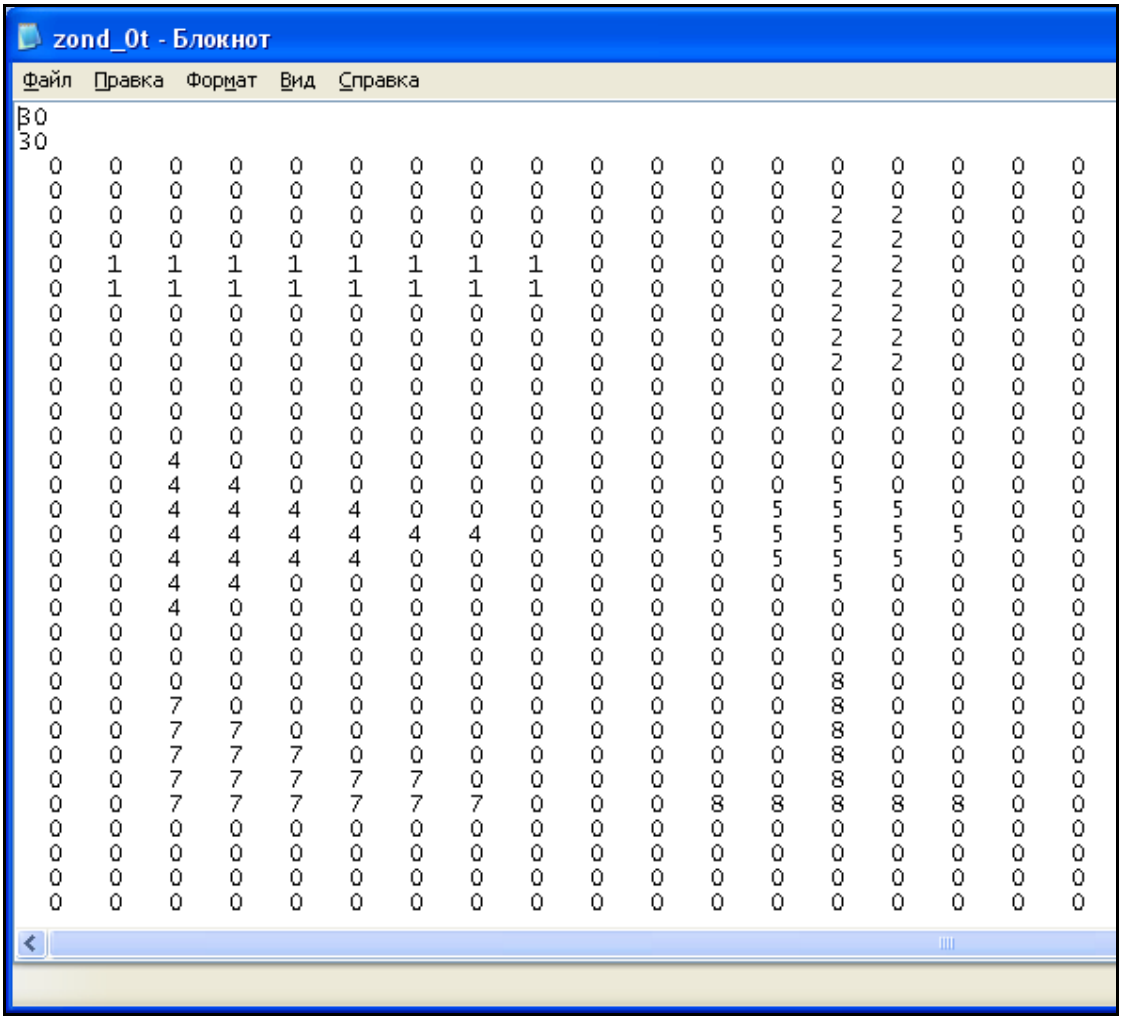


Рис. 2.1. Нумерація зондів у текстовому файлі

5. Для всіх класів символів C_k визначити еталонні значення станів зондів z і записати їх у масив $E[z,k]$ (рис. 1.4).
6. Створити власний варіант програми „*USB_Video_Rec_11V*”, виконавши в її коді такі зміни:
   - У процедурі „*p_Rec_Zond*” обчислити коефіцієнт подібності $K_C(T_i,C_k)$ зчитаного символу T_i з еталонами за формулою (1.2).
   - На основі максимального значення $K_C(T_i, C_k)$ визначити, до якого класу C_k належить зчитаний символ.
   - Розпізнавання зображення символу виконати для 4-х кутів повороту з кроком 90º. Відобразити на формі програми результати розпізнавання для такого кута повороту, при якому отримано максимальне значення $K_C(T_i, C_k)$.
7. Виконати розпізнавання зображень усіх символів згідно з варіантом за допомогою програми „*USB_Video_Rec_11V*”. Результати розпізнавання оформити у вигляді таблиці, в якій показати зображення зчитаних символів і еталонів, розміщення зондів на зображеннях символів, стани зондів для зчитаних символів і еталонів, значення коефіцієнта $K_C(T_i, C_k)$.

8. Розглянути будову і функціонування програми „*USB_Video_Rec_11V*” на рівні програмного коду. Створити спрощену схему алгоритму розпізнавання зображень символів за методом зондів (оператори візуалізації та роботи з файлами не враховувати).
9. *Додаткові завдання*.
   - У програмі „*USB_Video_Rec_11V*” при накладанні зондів на зображення символу виконувати зсуви зображення символу на 3 пікселі за шириною і висотою (з кроком у 1 піксель) для кращого суміщення зондів із символом.
   - При сегментації зображення символу змінювати значення порогу для кращого відтворення форми символу.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. У чому полягає задача розпізнавання образів? Що таке клас образів?
2. Сформулюйте математичну постановку задачі розпізнавання образів.
3. За яким принципом вибираються ознаки образу?
4. Наведіть приклад розпізнавання образів за набором ознак.
5. У чому полягає попередня обробка зображень перед їх розпізнаванням?
6. Перелічіть основні методи розпізнавання образів.
7. У чому полягає принцип шаблонних методів розпізнавання?
8. Поясніть принцип розпізнавання зображень методом зондів.
9. Яким способом можна отримати зображення кадру з відеопотоку?
10. Як змінити частоту зчитування кадрів відеопотоку в програмі „*USB_Video_Rec_11*”?
11. Як отримати RGB-складові кольору пікселів зображення?
12. Як виконується сегментація зображень у програмі „*USB_Video_Rec_11*”?
13. Запишіть формули, за якими виконується поворот зображення.
14. Як можна використовувати відеокамеру в якості детектора руху?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : КМ Академія, 2002. – 336 с.
2. Кутковецький В. Я. Розпізнавання образів : навчальний посібник / В. Я. Кутковецький. – Миколаїв : Вид-во МДГУ ім. П. Могили, 2003. – 196 с.

3. Люгер Дж. Ф. Искусственный интеллект: Стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2003. – 864 с.
4. Васильев В. И. Искусственный интеллект / В. И. Васильев, А. И. Шевченко. – Донецк : ДонДИИИ, 2000. – 360 с.
5. Форсайт Д. Компьютерное зрение. Современный подход / Д. Форсайт, Ж. Понс. – М. : Вильямс, 2004. – 928 с.
6. Агуров П. В. Интерфейсы USB. Практика использования и программирования / П. В. Агуров. – СПб. : БХВ-Петербург, 2004. – 576 с.

ЛАБОРАТОРНА РОБОТА № 2

КОМП'ЮТЕРНА РЕАЛІЗАЦІЯ ФАТИЧНОГО ДІАЛОГУ

Мета: ознайомити студентів із комп'ютерною реалізацією фатичного діалогу.

Завдання: програмно реалізувати фатичний діалог, вибравши його тематику згідно з варіантом.

Обладнання: персональний комп'ютер.

Програмне забезпечення: програма „*ELIZA*", середовище розробки програм *Delphi*.

1.ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Тест Тьюринга і фатичний діалог

Відомий англійський учений А. Тьюринг сформулював тезу, що визначає момент, з якого машину можна вважати інтелектуальною. Нехай експерт за допомогою телефону або подібного віддаленого пристрою спілкується з об'єктом, що може бути як людиною, так і машиною. Експерт дає певні тести-завдання. За результатами відповідей він повинен визначити, з ким має справу – з людиною чи комп'ютером. Якщо він приймає комп'ютер за людину, то комп'ютер може вважатися інтелектуальним. Така перевірка отримала назву тесту Тьюринга [1 - 4]. Багато спеціалістів вважали, що подібного тесту цілком достатньо для визначення рівня інтелектуальності комп'ютерної системи. Але виявилося, що це зовсім не так.

В основі тесту Тьюринга лежить неявне припущення про те, що необхідною умовою ведення діалогу є розуміння співрозмовника. Але наприкінці 60-х років американський кібернетик Дж. Вейценбаум створив дві програми – ЕЛІЗА і ДОКТОР – для використання в психіатрії. Виявилося, що вони можуть неправильно зорієнтувати експерта в умовах тесту Тьюринга. Тому при побудові таких програм в основу була покладена ідея фатичного діалогу.

Фатичним називається діалог без розуміння співрозмовника, на основі формального перефразування почутого. Наприклад, на фразу «Я хотів би поїхати у Львів» можна відповісти «Чому Ви хотіли б поїхати у Львів?». У ряді випадків програма може видати заздалегідь заготовлену фразу. Так, на слова експерта „Я знайшов помилку в програмі" можна відповісти „Всі можуть помилятися".

Такий діалог, звичайно, не має відношення до розуміння співрозмовника, але багато людей, з якими проводили фатичний діалог, вирішили, що мають справу з людиною. У [4] наводиться

курйозний приклад одного з перших фатичних діалогів. Віце-президент великої американської компанії захотів зв’язатися з відомим системним програмістом Д. Бобровим. Відбулася така телетайпна розмова:

Віце-президент: Чи можу я зараз скористатися комп’ютером?
Бобров: А чому Ви питаєте?
Віце-президент: Може, мені вдасться підписати контракт.
Бобров: Чому Ви в цьому впевнені?
Віце-президент: Мої потенційні покупці ніколи не бачили систему в дії.
Бобров: Це має для вас велике значення?
Віце-президент: Звичайно.
Бобров: Ви маєте в цьому сумнів?
Віце-президент: Що за безглузде зауваження?
Бобров: Що ви маєте на увазі, кажучи про безглузде зауваження?
Віце-президент: Чи можу я зараз скористатись комп’ютером?
Бобров: Розкажіть мені ще про Ваших потенційних покупців.
Віце-президент: Передзвоніть мені 491-1850.

Віце президент був здивований ще більше, коли йому ніхто не зателефонував. Виявилося, що телетайп з’єднував віце-президента не із самим Бобровим, а з програмою, яка реалізувала фатичний діалог.

Звичайно, подібний діалог важко вважати осмисленим, але дати чітке визначення рівня осмисленості також важко. Фатичний діалог може бути і менш примітивним, але системи, які здатні тільки на підтримку такого діалогу, не можна вважати інтелектуальними. Розуміння співрозмовника абсолютно незамінне для нормального діалогу, а для такого розуміння необхідно мати певну суму знань про світ.

1.2. Метод комп’ютерної реалізації фатичного діалогу

В основі псевдоінтелектуальних програм, які здатні підтримувати фатичний діалог із людиною або іншими аналогічними програмами, можуть лежати прості алгоритми зіставлення зі зразками. Наприклад, фатичний діалог із відомою програмою „ДОКТОР” може бути таким:

Доктор: Я Вас слухаю.
Людина: Мене бентежить погода.
Доктор: Як давно Вас бентежить погода?
Людина: Відтоді, як моя сім’я переїхала.
Доктор: Розкажіть про Вашу сім’ю.
Людина: У нас в сім’є всі захоплюються комп’ютерами.

Доктор: Машини Вас лякають.

Людина: Ні.

Доктор: Будь ласка, не будьте такі небагатослівні...

І так далі до безкінечності.

Таким чином, для реалізації діалогу потрібно використати механічне порівняння речень, які вводяться користувачем, зі зразками (шаблонами) речень, що зберігаються програмою (у масиві). У програмі використати один із нижчеподаних варіантів порівнянь:

Варіант 1 (повний збіг). Якщо речення, що вводиться, повністю збігається з одним із зразків, то йому відповідає наперед підготована відповідь.

Наприклад:

Людина: Яка система числення використовується в комп'ютерах?

Програма: Використовується двійкова система числення.

або

Людина: Квадрат гіпотенузи дорівнює сумі квадратів катетів.

Програма: Так, але є подібний результат і для непрямокутних трикутників – це теорема косинусів.

Повні збіги, звичайно, трапляються дуже рідко, тому використовуються інші типи порівнянь.

Варіант 2 (використання замінювачів). Типовим є використання замінювачів * і ?. Із замінювачем * зіставляється довільний фрагмент тексту, із замінювачем ? – будь-яке окреме слово.

Наприклад, шаблон (**комп'ютери**) успішно зіставляється з будь-яким реченням, в якому згадується про комп'ютери; шаблон (Я люблю ? яблука) – з такими реченнями, як (Я люблю солодкі яблука) тощо, але не з реченням (Я люблю їсти зелені яблука).

Варіант 3 (надання значень змінним «?» у процесі зіставлення). При цьому можливості програми, що реалізує фатичний діалог, значно розширюються. Вона набуває здатності до генерації відповідей, які залежать від запитань. Так, правило

(Я ? *) →(Що Ви ще ?)

дозволяє на речення (Я люблю яблука) відповісти (Що Ви ще любите?), а на речення (Я ненавиджу дощі) – (Що Ви ще ненавидите?). Безумовно, при використанні українських фраз потрібно стежити за узгодженням суфіксів і закінчень.

Варіант 4 (універсальний зразок). Зі зразком (*) зіставляється будь-яке речення. Звичайно, і відповіді, що відповідають цьому зразкові, повинні бути такими ж універсальними.

Наприклад: (Я Вас не дуже розумію); (Не будьте такими небагатослівними); (Чому це має для Вас значення?).

Варіант 5 (зіставлення більш ніж з одним зразком). Речення може зіставлятися не з одним зразком, а з кількома. Наприклад, якщо в програмі задані підстановки

(*люблю*) → (Що Ви ще любите?)

та

(*комп’ютери*) → (Ви маєте здібності до техніки),

то речення (Я люблю комп’ютери) зіставляється з обома зразками (випадковим чином вибирається один зі зразків).

Прикладом програми, яка реалізує фатичний діалог, є „ELIZA” (програма вільно розповсюджується), створена в середовищі Delphi на основі однойменної програми Дж. Вейценбаума (рис. 1.1).

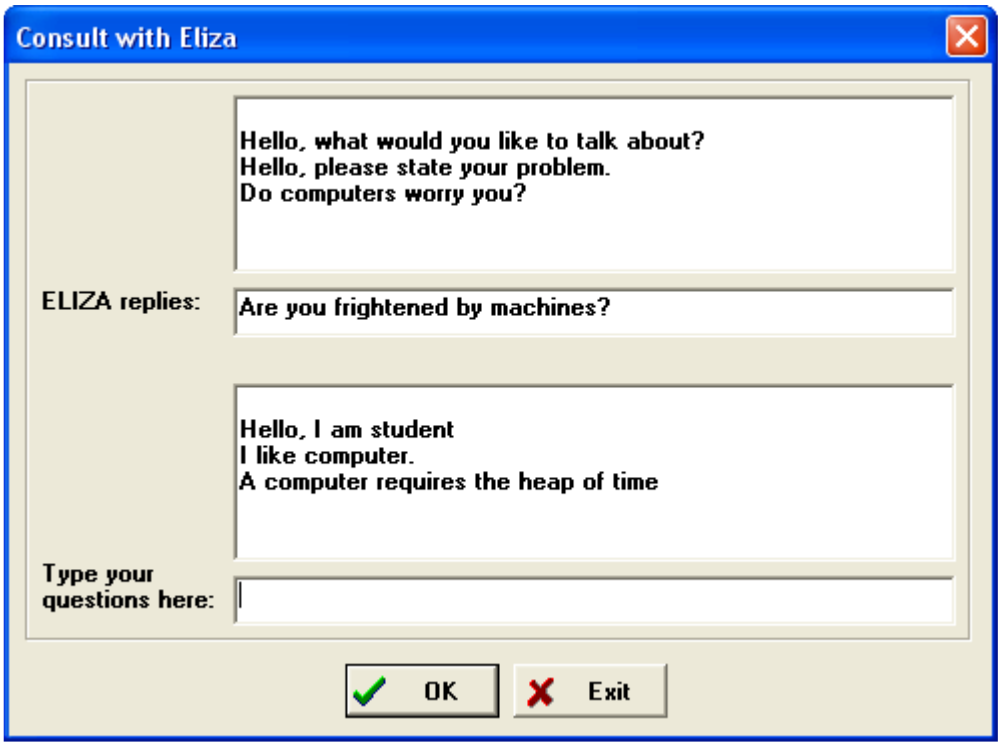


Рис. 1.1. Екранна форма програми „ELIZA” з фатичним діалогом

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитися з теоретичними відомостями.
2. Створити в середовищі Delphi програму, яка реалізує фатичний діалог. Програма повинна працювати так: користувач записує у вікні введення на формі речення, на яке програма повинна відповісти (одним реченням). Діалоги повинні бути виконані за певною тематикою (згідно з варіантами, табл. 2.1).
3. До звіту додати текст створеної програми з коментарями.

4. Додати до звіту приклади фатичного діалогу (10 запитань і відповідей).

Таблиця 2.1

Теми діалогів

Варіант	Тема	Варіант	Тема	Варіант	Тема
1	Автоматика	11	Комп'ютерні ігри	21	Процесори
2	Бази знань	12	Комп'ютерні мережі	22	Програмні агенти
3	Біоніка	13	Лінгвістика	23	Сканери
4	Графічні редактори	14	Математика	24	Спорт
5	Економіка	15	Медицина	25	Телевізори
6	Електронні таблиці	16	Мови програмування	26	Текстові редактори
7	Інтернет	17	Операційні системи	27	Фізика
8	Кібернетика	18	Монітори	28	Фрукти
9	Комп'ютерна графіка	19	Периферійні пристрої	29	Хімія
10	Комп'ютерні віруси	20	Персональні комп'ютери	30	Цифрові технології

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Охарактеризуйте поняття фатичного діалогу.
2. У чому полягає тест Тьюринга?
3. Як можна запрограмувати фатичний діалог?
4. Які можливі способи порівнянь речення, що вводиться при фатичному діалозі, із шаблонами?
5. Як використовуються замінювачі „*" і „?" у програмах, що реалізують фатичний діалог?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : КМ Академія, 2002. – 336 с.
2. Люгер Дж. Ф. Искусственный интеллект: Стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2003. – 864 с.
3. Поспелов Д. А. Фантазия или наука: На пути к искусственному интеллекту / Д. А. Поспелов. – М. : Наука, 1982. – 224 с.
4. Смолин Д. В. Введение в искусственный интеллект / Д. В. Смолин. – М. : Физматлит, 2004. – 208 с.

ЛАБОРАТОРНА РОБОТА № 3

НАВЧАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ МЕТОДОМ ЗВОРОТНОГО РОЗПОВСЮДЖЕННЯ ПОМИЛКИ

Мета: вивчити принципи побудови та навчання нейронних мереж, зокрема методом зворотного розповсюдження помилки.

Завдання: згідно з варіантом створити зображення символів, сформувати начальну і тестову вибірку для нейронної мережі, провести навчання нейромережі методом зворотного розповсюдження помилки та розпізнавання зображень символів за її допомогою.

Обладнання: персональний комп'ютер.

Програмне забезпечення: система *MatLab* із пакетом *Neural Network Toolbox*, програми „*Neuro_Image_Train*", „*Image_Neuro_Rec*".

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Штучні та біологічні нейронні мережі

Про роботу мозку в даний час відомо дуже мало, тому *штучні нейронні мережі* /ШНМ/ (*neural networks*) є надзвичайно спрощеною моделлю біологічних нейронних мереж. Особливість нейромереж (*neuronet*) – те, що їх *навчають*, а не програмують.

1.2. Штучний нейрон

Основними компонентами нейромережі є *нейрони* /*neurons*/ (елементи, вузли), які з'єднані зв'язками. Сигнали передаються по *зважених зв'язках* (*connection*), із кожним з яких пов'язаний *ваговий коефіцієнт* (*weighting coefficient*) або вага [1 - 7].

Штучний нейрон (ШН) імітує в першому наближенні властивості біологічного нейрона. Множина вхідних сигналів (вектор $X = \{x_1, x_2,\ldots, x_n\}$) надходить на ШН (рис. 1.1). Кожен сигнал множиться на відповідну вагу $w_1, w_2,\ldots, w_n$ і подається на суматор Σ (тіло біологічного нейрона). Вага w_i відповідає «силі» одного біологічного синаптичного зв'язку (множина ваг – вектор *W*). Суматор складає зважені входи алгебраїчно, створюючи вихід $NET = X \cdot W$.

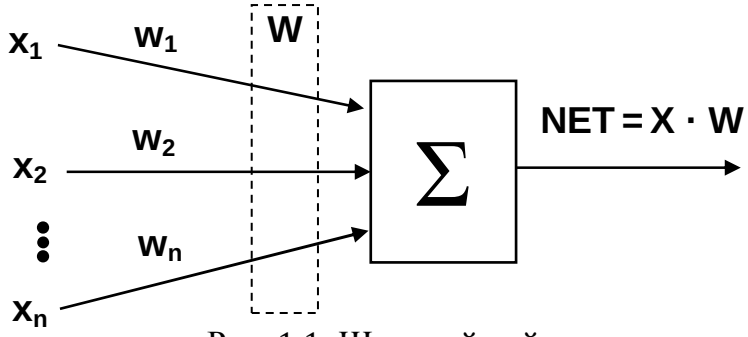


Рис. 1.1. Штучний нейрон

1.3. Активаційні функції нейронів

Сигнал *NET* далі, як правило, перетворюється активаційною функцією *F* і дає вихідний нейронний сигнал *OUT* = *F*(*NET*). Активаційна функція *F*(*NET*) може бути такою [6, 7]:

1. *Пороговою бінарною* функцією (рис. 1.2, *а*)

$$OUT = \begin{cases} 1, NET \geq T \\ 0, NET < T \end{cases}, \tag{1.1}$$

де *T* – деяка постійна порогова величина, або ж функція, що точніше моделює нелінійну передавальну характеристику біологічного нейрона.

2. *Лінійною обмеженою* функцією (рис. 1.2, *б*)

$$OUT = \begin{cases} 1, NET \geq T \\ NET, 0 \leq NET < T \\ 0, NET < 0 \end{cases}. \tag{1.2}$$

3. Функцією *гіперболічного тангенса* (рис. 1.2, *в*)

$$\mathrm{OUT} = th(C \cdot NET) = \frac{\exp(C \cdot NET) - \exp(-C \cdot NET)}{\exp(C \cdot NET) + \exp(-C \cdot NET)}, \tag{1.3}$$

де *C* > *0* – коефіцієнт ширини сигмоїди по осі абсцис (звичайно *C*=1).

4. *Сигмоїдною* (*S*-подібною) або логістичною функцією (рис. 1.2, *г*)

$$\mathrm{OUT} = \frac{1}{1 + \mathrm{e}^{-C \cdot \mathrm{NET}}}. \tag{1.4}$$

З виразу для сигмоїда очевидно, що вихідне значення нейрона лежить у діапазоні [0, 1]. Популярність сигмоїдної функції зумовлюють такі її властивості:

- здатність підсилювати слабкі сигнали сильніше, ніж великі, і опиратися „насиченню” від потужних сигналів;
- монотонність і диференційованість на всій осі абсцис;
- простий вираз для похідної

$$F'(NET) = C \cdot F(NET) \cdot (1 - F(NET)), \tag{1.5}$$

що дає можливість використовувати широкий спектр оптимізаційних алгоритмів.

У штучному нейроні з активаційною функцією (рис. 1.3) блок, позначений *F*, приймає сигнал *NET* і видає сигнал *OUT*. Якщо блок *F* звужує діапазон зміни величини *NET* так, що при будь-яких значеннях *NET* значення *OUT* належать деякому кінцевому інтервалу, то *F* називається «стискаючою» функцією. Як «стискаюча» функція часто використовується сигмоїдна функція.

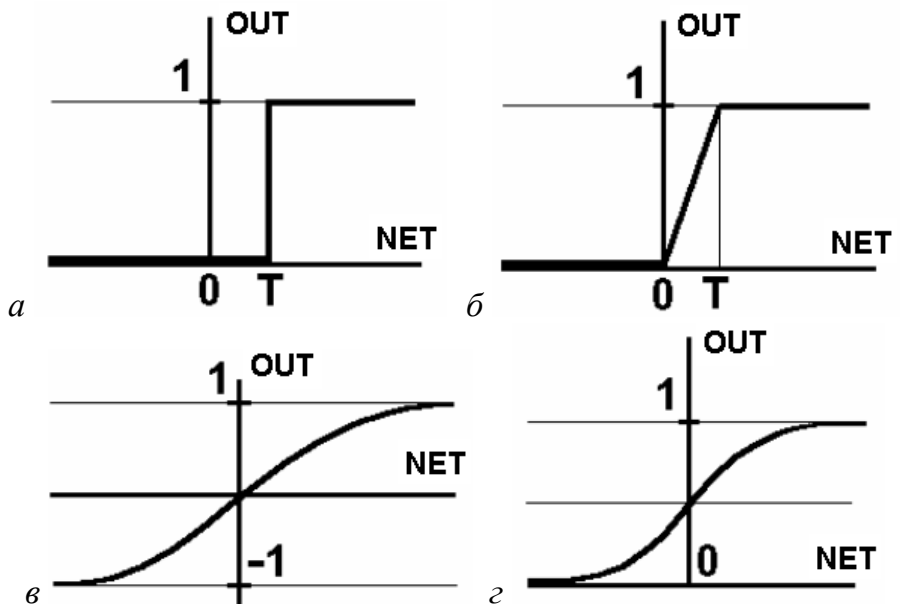


Рис. 1.2. Вид функцій активації: *а* – порогова бінарна функція; *б* – лінійна обмежена функція; *в* – гіперболічний тангенс; *г* – сигмоїд

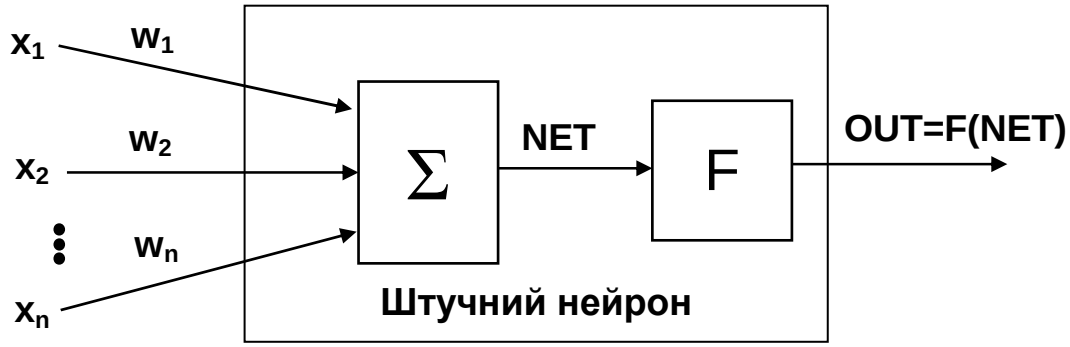


Рис. 1.3. Штучний нейрон з активаційною функцією *F*

1.4. Одношарові нейронні мережі

Хоча один нейрон і здатний виконувати прості процедури розпізнавання, ефективність нейронних обчислень залежить від з'єднань нейронів у мережі (рис. 1.4). Зазначимо, що вершини-круги зліва служать лише для розподілу вхідних сигналів і тому не вважатимуться шаром. Кожен елемент із множини входів *X* окремою вагою сполучений із кожним штучним нейроном.

Зручно вважати вагу елементами матриці *W*. Матриця має *n* рядків і *m* стовпців, де *n* – кількість входів, а *m* – кількість нейронів. Наприклад, $w_{3,2}$ – це вага, що пов'язує третій вхід із другим нейроном. Таким чином, обчислення вихідного вектора *Y*, компонентами якого є виходи *OUT* нейронів, зводиться до матричного множення $Y = X \cdot W$, де *Y* і *X* – вектори-рядки.

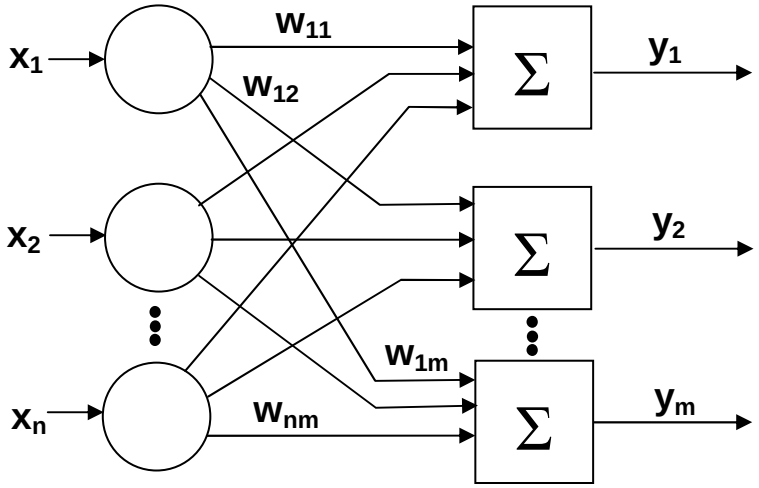


Рис. 1.4. Одношарова ШНМ

1.5. Персептрони і зародження штучних нейромереж

Персептрони складаються з одного шару штучних нейронів (рис. 1.5), сполучених за допомогою вагових коефіцієнтів із множиною входів (рис. 1.6).

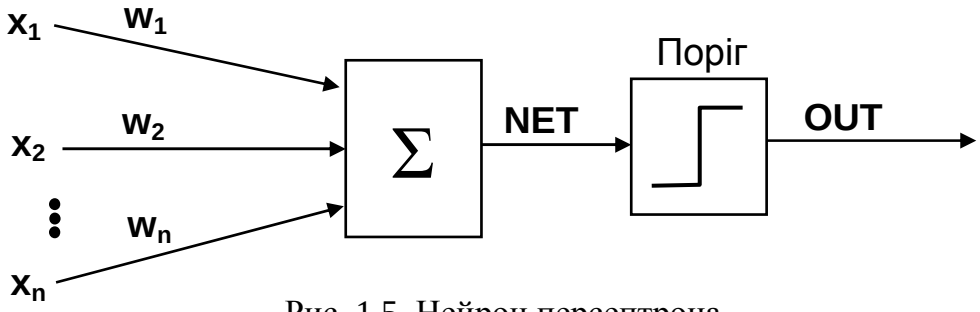


Рис. 1.5. Нейрон персептрона

Перші персептрони були створені Ф. Розенблатом у 60-х роках і викликали великий інтерес. Первинна ейфорія змінилася розчаруванням, коли виявилося, що персептрони не здатні навчитися розв’язанню ряду простих задач. Проте відкриття методів навчання багатошарових ШНМ сприяло відродженню інтересу до ШНМ.

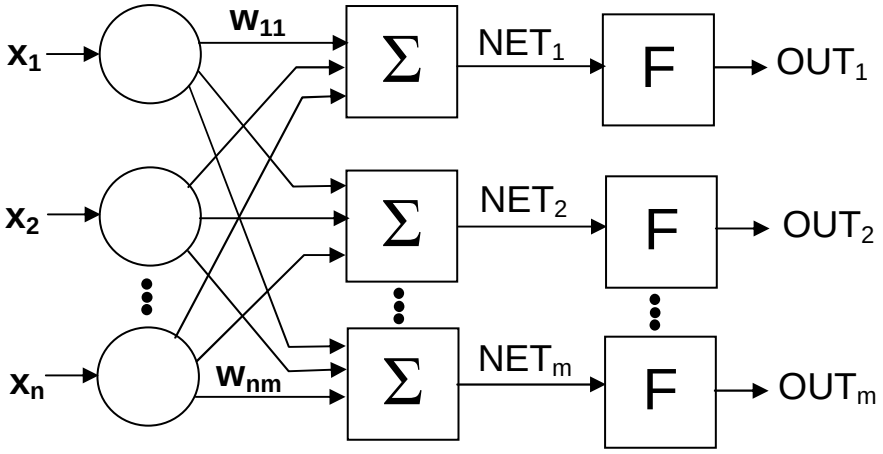


Рис. 1.6. Персептрон

1.6. Багатошарові нейронні мережі

Багатошарові мережі (рис. 1.7) володіють значно більшими можливостями, ніж одношарові. Проте багатошарові мережі можуть привести до збільшення обчислювальної потужності в порівнянні з одношаровими лише в тому випадку, якщо активаційна функція між шарами буде *нелінійною* (оскільки множення матриць асоціативне, то $X \cdot (W_1 \cdot W_2)$ еквівалентно $(X \cdot W_1) \cdot W_2$).

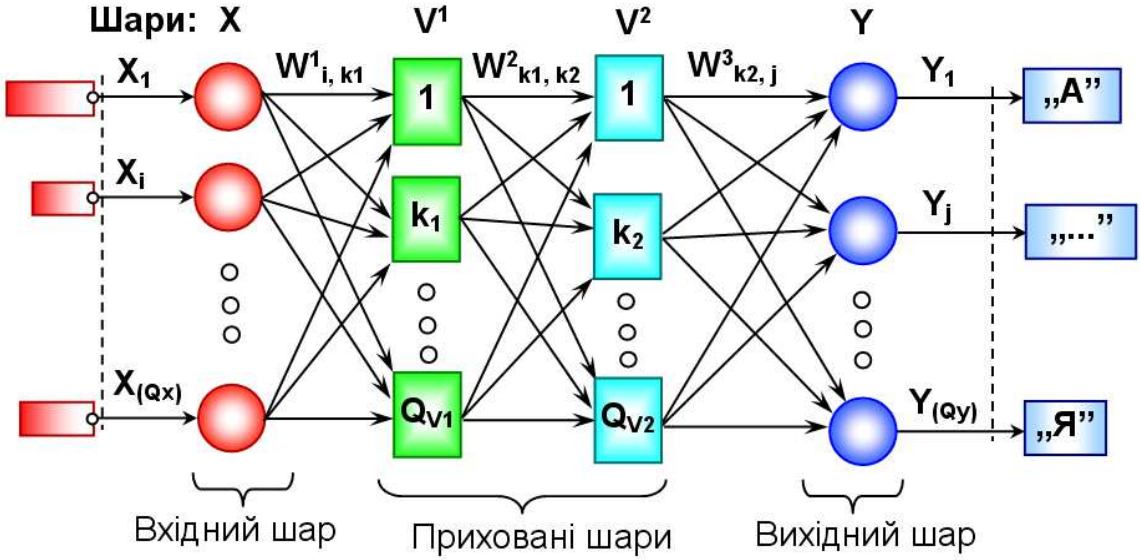


Рис. 1.7. Багатошарова ШНМ

1.7. Навчання нейронних мереж

Мережа навчається, щоб для деякої множини входів X давати бажану множину виходів Y. Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасним налагодженням ваг відповідно до певної процедури. В процесі навчання ваги мережі поступово стають такими, щоб кожен вхідний вектор виробляв потрібний вихідний вектор. Розрізняють алгоритми навчання з учителем і без учителя.

Навчання з вчителем передбачає, що для кожного вхідного вектора X існує цільовий вектор Y^T, що є необхідним виходом. Разом вони називаються навчальною парою. Звичайно мережа навчається для деякої кількості таких навчальних пар (навчальної множини). В ході навчання зчитується вхідний вектор X, обчислюється вихід мережі Y і порівнюється з відповідним цільовим вектором Y^T, різниця $D \sim Y^T - Y$ за допомогою зворотного зв'язку подається в мережу і змінюються ваги W відповідно до алгоритму, прагнучого мінімізувати помилку ε. Зчитування векторів навчальної множини й налагодження ваг виконується доти, поки сумарна помилка для всієї навчальної множини не досягне заданого низького рівня.

Навчання без вчителя. Незважаючи на численні прикладні досягнення, навчання з вчителем критикувалося за свою біологічну неправдоподібність. Важко уявити навчальний механізм у мозку, який би порівнював бажані й дійсні значення виходів, виконуючи корекцію за допомогою зворотного зв'язку. Якщо допустити подібний механізм в мозку, то звідки тоді виникають бажані виходи? Навчання без учителя – набагато правдоподібніша модель навчання в біологічній системі. Розвинена Кохоненом і багатьма іншими, вона не потребує цільового вектора для виходів і, отже, не вимагає порівняння з ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налагоджує вагу мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

1.8. Алгоритм зворотного розповсюдження помилки (backpropagation)

Зворотне розповсюдження помилки означає, що сигнали помилки з виходу мережі використовуються для корекції ваг попередніх шарів. Розглянемо алгоритм навчання на прикладі 3-шарової ШНМ (рис. 1.7), структура якої така:

1. *Вхідний* шар X; стани його елементів записані у векторі X_i, де $i = 1... Q_X$. Розмір навчальної множини (кількість векторів) на вході X дорівнює Qn, номер вектора $n = 1... Q_N$.

2. *Приховані* шари

Шар V^1 (рівень $L = 1$): вектор V^1_{k1}, де $k_1 = 1...Q_{V1}$; матриця ваг $W^1_{i,k1}$; різниця векторів D^1_{k1};

Шар V^2 ($L = 2$): вектор V^2_{k2}, де $k_2 = 1...Q_{V2}$; матриця ваг $W^2_{k1,k2}$; різниця D^2_{k2};

3. *Вихідний* шар ($L = 3$): стани його елементів записані у векторі Y_j; $j = 1...Q_Y$; матриця ваг $W^3_{k2,j}$; різниця D^3_j; істинний вихід (true) описується вектором Y^T_j, $j = 1...Q_Y$. Тобто Y^t_j – істинне значення для елемента j, Y_j – його реальних вихід.

ШНМ навчається за таким алгоритмом (рис. 1.8).

1. *Ініціалізація*. Початкові вагові коефіцієнти W встановлюються рівними малим випадковим значенням з діапазону [$-\Delta W$,.. ΔW] (наприклад, $\Delta W = 0.3$):

$$W^1_{i,k1} = (Rnd - 0{,}5) \cdot 2 \cdot \Delta W\ ,\ i = 1..qX\ ,\ k_1 = 1..qV_1.$$

У вагових матрицях рядки відповідають елементам, від яких йдуть зв'язки, а стовпці – до яких йдуть зв'язки.

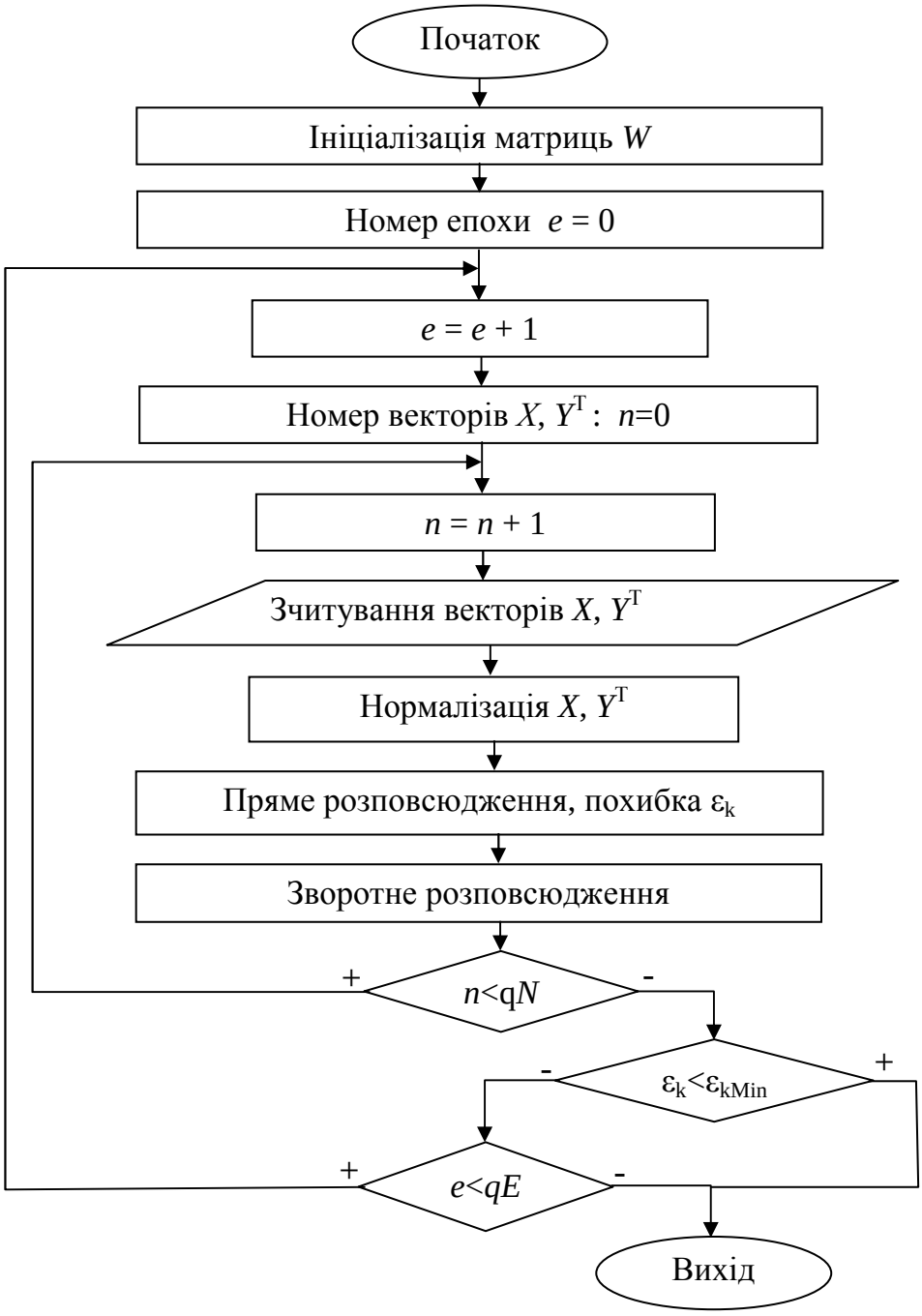


Рис. 1.8. Алгоритм навчання нейромережі зі зворотним розповсюдженням помилки

2. *Нормалізація* (масштабування) значень усіх векторів X, Y^T(для кожного типу окремо) в діапазон (*MinN*; *MaxN*), наприклад *MinN* = 0.01; *MaxN* = 0.99.

$$X_i^n = MinN + \frac{(X_i - X_{\min}) \cdot (MaxN - MinN)}{(X_{\max} - X_{\min})}.$$

3. *Пряме розповсюдження* (Direct) полягає у знаходженні вихідного вектора Y на основі вхідного X за такими формулами:

Шар 1: $Net_{k1} = \sum_{i=1}^{qX} X_i \cdot W^1_{i,k1}$, $V^1_{k1} = f\left(\frac{1}{1+\exp(-Net_{k1})}\right)$, де $k_1 = 1..qV_1$.

Шар 2: $Net_{k2} = \sum_{k1=1}^{qV1} V^1{}_{k1} \cdot W^2_{k1,k2}$, $V^2_{k2} = f\left(\frac{1}{1+\exp(-Net_{k2})}\right)$, де $k_2 = 1..qV_2$.

Шар 3: $Net_j = \sum_{j=1}^{qY} V^2{}_{k2} \cdot W^3_{k2,j}$, $Y_j = f\left(\frac{1}{1+\exp(-Net_j)}\right)$, де $j = 1..qY$.

У результаті прямого розповсюдження можна обчислити помилки навчання мережі (для всіх векторів навчальної множини):

- $\varepsilon = \sum_{n=1}^{qN} \sum_{j=1}^{qY} \left| Y_{j,n} - Y^T_{j,n} \right|$ – лінійна помилка;
- $\varepsilon_k = \frac{1}{qN} \frac{1}{qY} \sum_{n=1}^{qN} \sum_{j=1}^{qY} \left(Y_{j,n} - Y^T_{j,n} \right)^2$ – середня квадратична помилка (mse – Mean squared error);
- $\varepsilon_\sigma = \frac{1}{qN \cdot qY} \sum_{n=1}^{qN} \sum_{j=1}^{qY} \left(\frac{\left| Y_{j,n} - Y^T_{j,n} \right|}{\left| Y^T_{j,n} \right| + 1} \right) \cdot 100\%$ – середня відносна помилка (0 – 100%).

Якщо значення сумарної квадратичної помилки ε_k менше від заданого, то процес навчання мережі завершується.

4. *Зворотне розповсюдження* помилки (*backpropagation*) полягає в корекції вагових коефіцієнтів через сигнал різниці *D*.

Шар 3: $D^3_j = Y_j \cdot (1 - Y_j)(Y^t_j - Y_j)$, $W^{3(e)}_{k2,j} = W^{3(e-1)}_{k2,j} + \eta_Y \cdot D^3_j \cdot V^2_{k2}$,

де $j = 1..qY$, e – номер епохи (*epochs*); оскільки як активаційна функція використовується сигмоїдна, тому різниця векторів (Y^T-Y) множиться на похідну від сигмоїдної функції: $Y(1 - Y)$.

Шар 2: $D^2_{k2} = \sum_{j}^{qY} V^2_{k2} \cdot (1 - V^2_{k2}) \cdot (D^3_j \cdot W^3_{k2,j})$, $W^{2(e)}_{k1,k2} = W^{2(e-1)}_{k1,k2} + \eta_{L2} \cdot D^2_{k2} \cdot V^1_{k1}$,

де $k_2 = 1..qV_2$.

Шар 1: $D^1_{k1} = \sum_{k2}^{qV2} V^1_{k1} \cdot (1 - V^1_{k1}) \cdot (D^2_{k2} \cdot W^2_{k1,k2})$, $W^{1(e)}_{i,k1} = W^{1(e-1)}_{i,k1} + \eta_{L1} \cdot D^1_{k1} \cdot X_i$,

де $k_1 = 1..qV_1$, η_Y, η_{L2}, η_{L1} – норми навчання (наприклад, 0.5).

Із метою контролю значень матриці *W* визначаються: *Min* – мінімальне значення; *Max* – максимальне; *Ms* – математичне сподівання; *Sigma* – середньоквадратичне відхилення.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Згідно з номером варіанта (*V*) створити 4 початкові чорно-білі /бітові/ зображення (еталони) друкованих символів (табл. 2.1) розміром 16*16 пікселів (зображення зберегти у файлах формату *.bmp). Назви файлів початкових зображень вибрати відповідно до назви символів (наприклад, файл *A1.bmp* для символу „*A*”).

Для кожного початкового зображення створити такі зображення розміром 16 × 16 пікселів (рис. 2.1):

- зміщене / *Shift* / (назва файла, наприклад, A1_S1.bmp);
- повернуте / *Rotate* / (назва файла, наприклад, A1_R1.bmp);
- пошкоджене / *Damage* / (назва файла, наприклад A1_D1.bmp);
- масштабоване /*sCale* / (назва файла, наприклад A1_C1.bmp).

а *б* *в* *г*

Рис. 2.1. Зображення символу „А”: *а* – початкове; *б* – зміщене; *в* – повернуте; *г* – пошкоджене

Зміни зображень символів можна виконати за допомогою, наприклад, програм *Paint* та *Adobe Photoshop*.

2. Створити навчальну вибірку нейромережі – отримані 20 зображень записати в папку „*Neuro_Train*”, розміщену в папці головної програми (рис. 2.2).

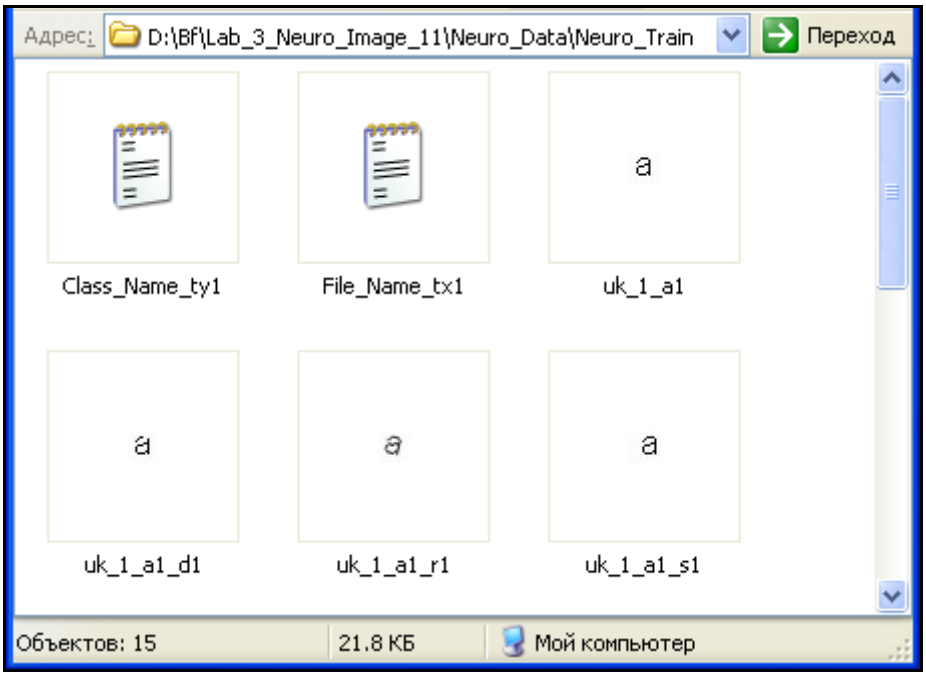
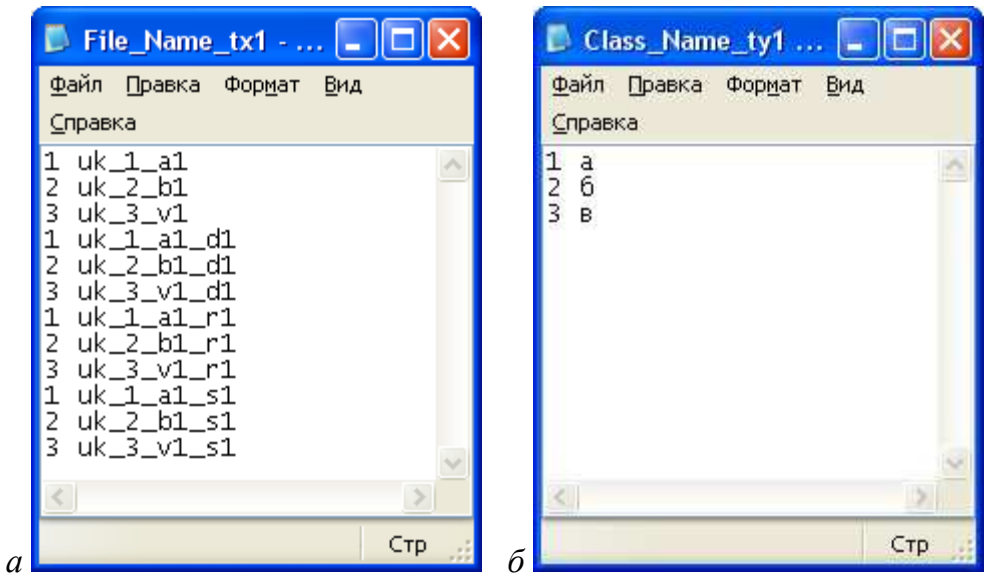


Рис. 2.2. Фрагмент навчальної (train) вибірки ШНМ

Перелік зображень навчальної вибірки записати у файл „*File_Name_tx1.txt*” (рис. 2.3, а). У файл назв класів “*Class_Name_ty1.txt*” (рис. 2.3, б) записати назви символів згідно з варіантом, при цьому назви класів пронумерувати (починаючи з 1 із кроком 1). У файлі навчальної вибірки (рис. 2.3, а) перед назвою кожного зображення записати номер відповідного класу (рис. 2.3, б). У вищеописаних файлах у кожному рядку номер і назва повинні розділятися одним пробілом.



а *б*

Рис. 2.3. Структура файла навчальної вибірки (*а*) та файла назв класів (*б*)

3. Створити тестову вибірку нейромережі з 16-ти зображень. Для цього на основі кожного початкового зображення потрібно створити 4 змінених зображення (аналогічно як для навчальної вибірки, але величина трансформації не повинна бути не більшою, ніж для відповідного зображення навчальної вибірки). Отримані 16 змінених зображень записати в папку „*Neuro_Recogn*”, розміщену в папці головної програми (рис. 2.4).

Рис. 2.4. Фрагмент тестової вибірки ШНМ

4. Виконати навчання 3-шарової ШНМ за допомогою програми „*Neuro_Image_Train*” у системі Matlab (лістинг 2.1) [8; 9]. Для цього потрібно згідно з варіантом (табл. 2.2) вибрати активаційні функції нейронів прихованих шарів (розділ „*s*2” програми) й алгоритм навчання нейромережі (розділ „*s*4” програми). Для непарних варіантів створити класичну нейромережу з навчанням методом зворотного розповсюдження помилки, а для парних – каскадну нейромережу з навчанням методом зворотного розповсюдження помилки (розділ „*s*2” програми). Навчання провести для таких значень мінімальної помилки *mse* (*net.trainParam.goal*, розділ „*s*5” програми): 10^{-2}, 10^{-3}, 10^{-4} (всі інші параметри – за замовчуванням); для кожного випадку записати час навчання мережі, навчені ШНМ зберегти у файл.

5. Виконати розпізнавання 16-ти зображень створених символів із папки „*Neuro_Recogn*” за допомогою ШНМ програмою „*Neuro_Image_Recog*”, використовуючи нейромережі, отримані для різних значень похибки *mse*. Для кожного зображення записати отриманий вихідний вектор *Y* з командного вікна Matlab і результат розпізнавання (оформити у вигляді таблиці звіту). Вектори *Y* показати також у вигляді діаграм. На основі отриманих даних зробити висновки про ефективність розпізнавання. Які зображення розпізнаються краще?

Таблиця 2.1

Символи

Варіант	Символ					Варіант	Символ			
V	1	2	3	4		V	1	2	3	4
1	α	е	G	Е		16	π	с	V	С
2	β	є	H	Є		17	ρ	т	W	Т
3	γ	ж	I	Ж		18	ς	у	X	У
4	δ	з	J	З		19	σ	ф	Y	Ф
5	ε	и	K	И		20	τ	х	Z	Х
6	ζ	і	L	І		21	υ	ц	Э	Ц
7	η	ї	M	Ї		22	φ	ч	Ы	Ч
8	θ	й	N	Й		23	χ	ш	6	Ш
9	ι	к	O	К		24	ψ	щ	7	Щ
10	κ	л	P	Л		25	ω	ю	8	Ю
11	λ	м	Q	М		26	β	я	9	Я
12	μ	н	R	Н		27	γ	ь	0	Ь
13	ν	о	S	О		28	δ	б	D	Б
14	ξ	п	T	П		29	ε	г	E	Г
15	о	р	U	Р		30	ζ	д	F	Д

Таблиця 2.2

Активаційні функції нейронів та алгоритми навчання нейромережі

Варіант V	Активаційні функції	Алгоритм навчання
1	сигмоїдна, сигмоїдна	trainbfg, квазіньютонівський зворотного поширення помилки
2	сигмоїдна, сигмоїдна	traingd, градієнтного спуску
3	сигмоїдна, сигмоїдна	traingda, градієнтного спуску з параметром швидкості
4	сигмоїдна, сигмоїдна	traingdm, градієнтного спуску зі збуренням
5	сигмоїдна, сигмоїдна	trainrp, пороговий алгоритм зворотного поширення помилки
6	сигмоїдна, сигмоїдна	traincgf, Флетчера – Пауела
7	сигмоїдна, сигмоїдна	traincgp, градієнтного спуску Полака – Рибейри
8	сигмоїдна, сигмоїдна	trainscg, градієнтного спуску Моллера
9	сигмоїдна, сигмоїдна	trainoss, методу січної
10	сигмоїдна, сигмоїдна	trainlm, Левенберга – Марквардта
11	сигмоїдна, тангенс гіперболічний	trainbfg, квазіньютонівський зворотного поширення помилки
12	сигмоїдна, тангенс гіперболічний	traingd, градієнтного спуску
13	сигмоїдна, тангенс гіперболічний	traingda, градієнтного спуску з параметром швидкості
14	сигмоїдна, тангенс гіперболічний	traingdm, градієнтного спуску зі збуренням
15	сигмоїдна, тангенс гіперболічний	trainrp, пороговий алгоритм зворотного поширення помилки
16	сигмоїдна, тангенс гіперболічний	traincgf, Флетчера – Пауела
17	сигмоїдна, тангенс гіперболічний	traincgp, градієнтного спуску Полака – Рибейри
18	сигмоїдна, тангенс гіперболічний	trainscg, градієнтного спуску Моллера
19	сигмоїдна, тангенс гіперболічний	trainoss, методу січної
20	сигмоїдна, тангенс гіперболічний	trainlm, Левенберга – Марквардта

Продовження таблиці 2.2

Активаційні функції нейронів та алгоритми навчання нейромережі

Варіант V	Активаційні функції	Алгоритм навчання
21	сигмоїдна, сигмоїдна	trainbfg, квазіньютонівський зворотного поширення помилки
22	сигмоїдна, сигмоїдна	traingd, градієнтного спуску
23	сигмоїдна, сигмоїдна	traingda, градієнтного спуску з параметром швидкості
24	сигмоїдна, сигмоїдна	traingdm, градієнтного спуску зі збуренням
25	сигмоїдна, сигмоїдна	trainrp, пороговий алгоритм зворотного поширення помилки
26	сигмоїдна, сигмоїдна	traincgf, Флетчера – Пауела
27	сигмоїдна, сигмоїдна	traincgp, градієнтного спуску Полака – Рибейри
28	сигмоїдна, сигмоїдна	trainscg, градієнтного спуску Моллера
29	сигмоїдна, сигмоїдна	trainoss, методу січної
30	сигмоїдна, сигмоїдна	trainlm, Левенберга – Марквардта

6. Провести навчання ШНМ для трьох випадків: при зміні кількості нейронів у прихованих шарах (розділ „*s*2” програми „*Neuro_Image_Train*”) на величину -50 %, 0% та +50 % (при $mse = 10^{-3}$). Для парних варіантів змінювати кількість нейронів у шарі 1 (Q_{V1}), а для непарних – у шарі 2 (Q_{V2}). Навчання ШНМ проводити $m = 7$ разів для кожного випадку, в результаті для кожного випробовування k отримати час навчання $t(k)$. Усі значення часу навчання оформити у вигляді таблиці та діаграм. Оскільки за рахунок випадкового вибору ваг результати навчання ШНМ будуть кожного разу відрізнятися, тому для їх обробки потрібно використати статистичні методи (див. додаток А).

3. ДОВІДКА ДЛЯ РОБОТИ З ПРОГРАМАМИ „NEURO_IMAGE_TRAIN” ТА „IMAGE_NEURO_REC”

Програма „*Image_Neuro_Study*” призначена для навчання ШНМ, а програма „*Image_Neuro_Rec*” виконує розпізнавання зображень за допомогою ШНМ. Навчена ШНМ зберігається у файлі. Програма „*Image_Neuro_Rec*” зчитує навчену ШНМ із файла й виконує розпізнавання символів на основі їх зображень. Програми створені в системі MatLab у вигляді m-файлів сценаріїв [8 - 10].

3.1. Створення штучних нейронних мереж у програмі „Image_Neuro_Study”

Для створення ШНМ у програмі використовуються функції *newcf* та *newff*. Ці функції моделюють каскадну і класичну багатошарові ШНМ із навчанням методом зворотного розповсюдження помилки. Як активаційні функції нейронів використовуються сигмоїдальні (*logsin*), функції гіперболічного тангенсу (*tansig*) та лінійні (*purelin*).

Розглянемо приклад створення штучної нейронної мережі *net*:

net = newff (P, T, [Q1 ... QL], {'tansig', ... 'logsin'}),

де *P* – матриця вхідних векторів навчальної вибірки (отримується за допомогою m-файла „*p_Read_Vector*_1”);

T – матриця цільових векторів навчальної вибірки;

L – кількість шарів ШНМ;

*Q*1, *QL* – кількості нейронів у першому та *L*-му прихованих шарах ШНМ;

'tansig', 'logsin' – активаційні функції для першого та *L*-го шару.

Матриця *P* вхідних векторів навчальної вибірки має таку структуру (3.1):

$$P = \begin{pmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,n} & \dots & P_{1,Q_N} \\ P_{2,1} & P_{2,2} & \dots & P_{2,n} & \dots & P_{2,Q_N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{i,1} & P_{i,2} & \dots & P_{i,n} & \dots & P_{i,Q_N} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{Q_X,1} & P_{Q_X,2} & \dots & P_{Q_X,n} & \dots & P_{Q_X,Q_N} \end{pmatrix}, \quad (3.1)$$

де n – номер зображення; Q_N – кількість елементів навчальної вибірки (зображень), тобто кількість вхідних векторів X.

У матриці P кожний стовпець є одним вектором X (3.2):

$$X = \begin{pmatrix} X_1 \\ \dots \\ X_i \\ \dots \\ X_{Q_X} \end{pmatrix}, \quad (3.2)$$

де Q_X – розмірність вхідного вектора (кількість ознак зображення, як ознаки зображення використовується яскравість його пікселів).

3.2. Навчання штучних нейронних мереж у програмі „Image_Neuro_Study”

Навчання ШНМ виконується функцією *train* за допомогою алгоритмів (табл. 2.2), які є модифікацією методу зворотного поширення помилки (рис. 3.1).

Розглянемо фрагмент коду, який реалізує навчання ШНМ *net*:

NET. trainFcn = 'trainbfg';
[net, tr] = train (net, P, T, Ai, Ts, VV, TV),

де NET. trainFcn – алгоритм навчання ШНМ;
P – матриця вхідних векторів навчальної вибірки;
T – матриця цільових векторів навчальної вибірки;
Ai – матриця початкових умов для вхідних затримок;
Ts – вектор часових інтервалів;
VV – матриця контрольної вибірки;
TV – матриця тестової вибірки.

Слід врахувати, що параметри *Ai, Ts, VV і TV* не обов’язкові.

Функція *train* повертає навчену ШНМ *net* і набір записів *tr* для кожного циклу навчання (епохи):

tr.epoch – номер епохи;
tr.perf – помилка навчання (mse).

Процес навчання виконується відповідно до наступних параметрів:

net.trainParam.epochs – задана кількість епох;
net.trainParam.goal – цільова помилка mse, при досягненні якої процес навчання завершується;
net.trainParam.time – максимальний час навчання в секундах;
net.trainParam.min_grad – мінімально допустимий градієнт mse.

Лістинг 1. Програма „*Neuro_Image_Train*”, яка виконує навчання штучної нейронної мережі

```
% s1 ------------------- Read Vector -------------------------
  % На початку роботи програми відкрити файл назв класів
  % Neuto_Train\Class_Name_ty1.txt
  p_Read_Vector_1;    % Зчитати матриці P та T
   % P - матриця вхідних векторів навчальної вибірки
   % T - матриця вихідних (цільових) векторів навчальної вибірки

% s2 -------------------- Create NetWork ----------------------
  %  Створити нейронну мережу із визначеною структурою
 net = newff(P,T,[16 12],{'tansig',' logsin'});
   % створити нейромережу функцією newff з 2 прихованими шарами,
   % кількість нейронів в яких дорівнює 16 і 12 відповідно,
```

```
   % і активіаційними функціями нейронів  tansig, logsin.
     %   Види активаційних функцій:
   % purelin – лінійна обмежена функція;
   % tansig – гіперболічний тангенс;
   % logsin – сигмоїдна функція
% newff – створити класичну нейромережу з навчанням методом
% зворотного розповсюдження помилки (feed-forward
% backpropagation network)
% newcf – створити каскадну нейромережу з навчанням методом
% зворотного розповсюдження помилки (cascade-forward
% backpropagation network)

 % s3 --------- NetWork gen Simulink -------------------------
      gensim(net);  %  Моделювати нейромережу в Simulink

 % s4 --------- Learning Algorithm of Network -----------------
  % Вибрати алгоритм навчання нейромережі:

% N_0.  BFGS quasi-Newton backpropagation
% Квазіньютонівський алгоритм зворотного поширення помилки
  %  NET.trainFcn='trainbfg';

% N_1. Gradient descent backpropagation
% Алгоритм градієнтного спуску
  %  NET.trainFcn='traingd';

% N_2. Gradient descent with adaptive backpropagation;  Алгоритм
% градієнтного спуску з параметром швидкості налаштування
    net.trainFcn = 'traingda';


% N_3. Gradient descent with momentum backpropagation
% Алгоритм градієнтного спуску зі збуренням
  %  net.trainFcn = 'traingdm';

% N_4. Resilient backpropagation
% Пороговий алгоритм зворотного поширення помилки
  %  net.trainFcn = 'trainrp';

% N_5. Fletcher – Powell conjugate gradient backpropagation
% алгоритм Флетчера – Пауела
  %  net.trainFcn = 'traincgf';

% N_6. Polak – Ribiere conjugate gradient backpropagation
%  Алгоритм градієнтного спуску Полака – Рибейри
  % net.trainFcn = 'traincgp';

% N_7. Scaled conjugate gradient backpropagation
% Алгоритм градієнтного спуску Моллера
   % net.trainFcn = 'trainscg';
```

```
 % N_8. One step secant backpropagation
% Алгоритм методу січної OSS
  %  net.trainFcn = 'trainoss';

% N_9. Levenberg – Marquardt backpropagation
% Алгоритм Левенберга – Марквардта
   % net.trainFcn = 'trainlm';

% s5 -------------------- Train Parameter ----------------------
 net.performFcn = 'mse';  %  mse - Mean squared error
  % mse - середньоквадратична помилка навчання нейромережі

net.trainParam.epochs=1500;  % Максимальна кількість епох
net.trainParam.goal=1E-4;        % Мінімальна помилка навчання
 net.trainParam.time=60;          % Максимальний час навчання

 % s6 ------------------- Train NetWork -----------------------
  [net,tr] = train(net,P,T,[],[],[],[]);   % виконати навчання нейромережі

  ep1=tr.epoch;  % Epoch number; номери епох
   mse1=tr.perf;   % Помилки навчання для кожної епохи

 % s7 ------------------- Save NetWork -----------------------
     save('net_1');   % зберегти навчену нейромережу у файл
```

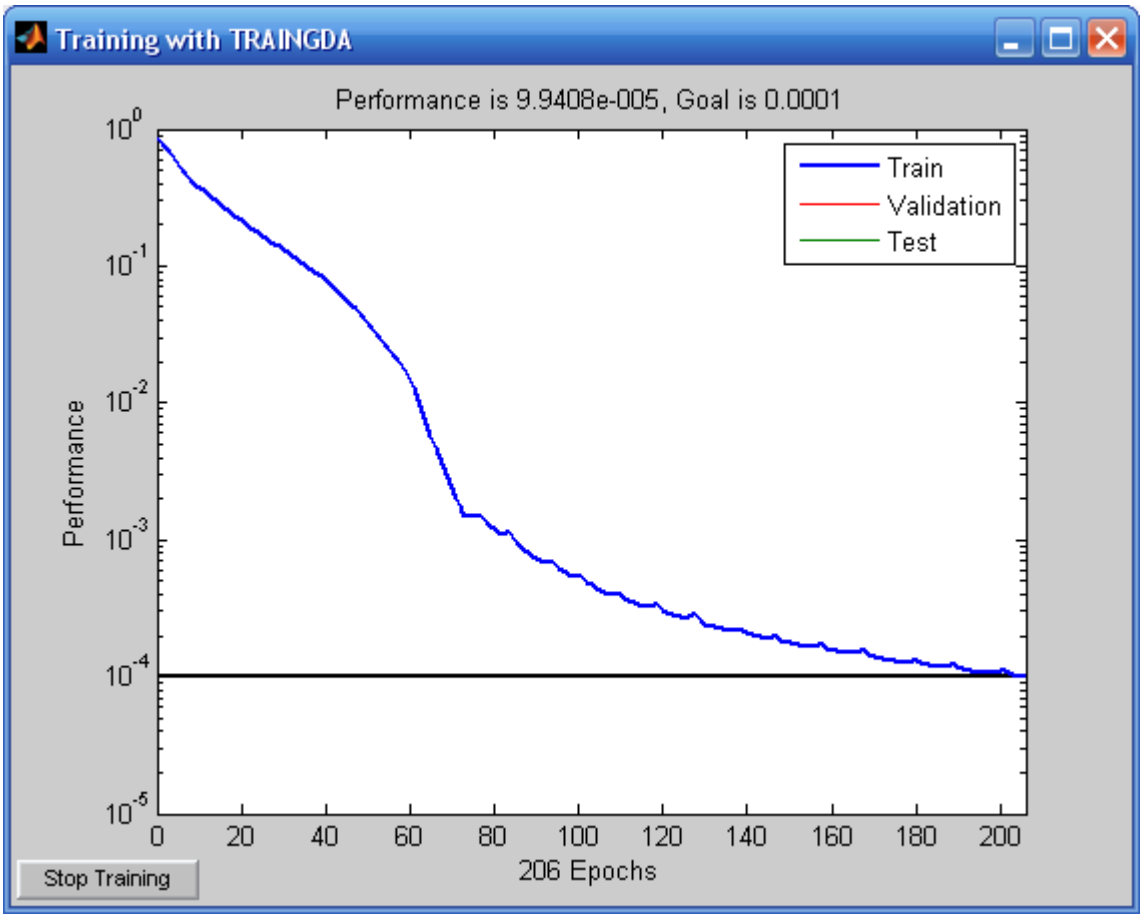


Рис. 3.1. Залежність помилки навчання mse від номера епохи

3.3. Візуалізація штучних нейронних мереж

У системі MatLab можлива візуалізація ШНМ засобами Simulink (рис. 3.2, рис. 3.3).

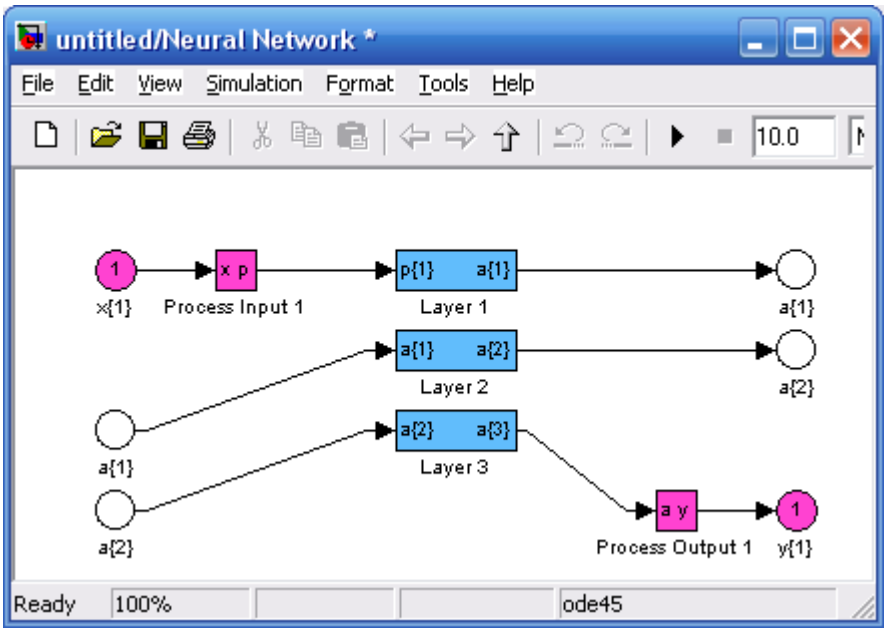


Рис. 3.2. Структура штучної нейронної мережі

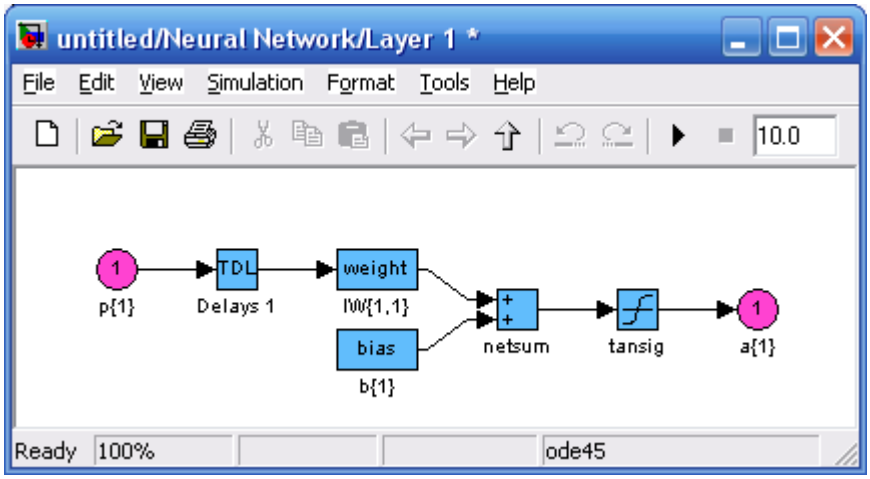


Рис. 3.3. Структура нейрона

Додаток А

Аналіз значень випадкової величини – часу навчання ШНМ $t(k)$ виконати за методом Монте – Карло. Для цього для кожного випадку (при зміні кількості нейронів у прихованих шарах на -50 %, 0% та +50 %) потрібно виконати такі дії:

1. Обчислити середнє значення часу навчання (математичне сподівання) за формулою

$$t^c = \frac{1}{m}\left(\sum_{k=1}^{m} t_k\right), \qquad \text{(A.1)}$$

і середнє квадратичне відхилення

$$\sigma = \sqrt{\frac{1}{m}\left(\sum_{k=1}^{m}\left(t_k - t^c\right)^2\right)}. \qquad \text{(A.2)}$$

2. Розрахувати верхню межу помилки δ визначення часу навчання НМ з надійністю γ для m випробувань

$$\delta = t_\gamma s / \sqrt{m}\ , \qquad \text{(A.3)}$$

де t_γ – значення аргументу функції Лапласа, при якому $\Phi(t_\gamma) = \gamma/2$;

γ – надійність верхньої межі помилки δ; $s = \sqrt{\frac{m}{m-1}} \cdot \sigma$.

Обчислення проводити для надійності γ = 0.95, якій при m = 7 відповідає значення t_γ = 2.45.

3. Обчислити найменшу кількість випробувань m_1, які забезпечують наперед задану верхню межу помилки δ_1 = 0.1 за формулою

$$m_1 = t_\gamma^2 s^2 / \delta_1^2 . \qquad \text{(3.6)}$$

Отримані значення t^c, σ, δ, m_1 оформити у вигляді таблиці.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Що таке штучний нейрон?
2. Опишіть спрощену будову та принцип дії біологічного нейрона.
3. У чому полягає роль вагових коефіцієнтів?
4. Перерахуйте та опишіть основні активаційні функції нейронів, наведіть відповідні формули і графіки.
5. Чому як активаційна часто використовується сигмоїдна функція?
6. Зобразіть структуру одношарової нейромережі, поясніть принцип її функціонування.
7. Що таке „персептрон”?
8. У чому полягає головний недолік персептрона?
9. Зобразіть структуру багатошарової нейромережі, поясніть принцип її функціонування.
10. Чому для ефективної роботи багатошарової нейромережі активаційна функція повинна бути нелінійною?
11. Опишіть основні способи навчання нейромереж, поясніть їх недоліки та переваги.

12. У чому полягає концепція Хебба, яка використовується при навчанні НМ?
13. Опишіть алгоритм навчання нейромережі зі зворотним розповсюдженням помилки.
14. Як формуються матриці вагових коефіцієнтів НМ?
15. Для чого потрібно проводити нормалізацію значень вхідних і вихідних векторів НМ?
16. Опишіть математичну модель прямого розповсюдження сигналу на прикладі 3-шарової нейромережі.
17. Як математично описуються помилки навчання нейромережі?
18. Опишіть математичну модель зворотного розповсюдження помилки на прикладі 3-шарової нейромережі.
19. Чому нейромережі ефективні при розпізнавані зображень?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : КМ Академія, 2002. – 336 с.
2. Заенцев И. В. Нейронные сети: основные модели / И. В. Заенцев. – Воронеж : Изд-во Воронеж. ун-та, 1999. – 76 с.
3. Каллан Р. Основные концепции нейронных сетей / Р. Каллан. – М. : Вильямс, 2003. – 288 с.
4. Кутковецький В. Я. Розпізнавання образів : навчальний посібник / В. Я. Кутковецький. – Миколаїв : Вид-во МДГУ ім. П. Могили, 2003. – 196 с.
5. Люгер Дж. Ф. Искусственный интеллект: Стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2003. – 864 с.
6. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия – Телеком, 2004. – 452 с.
7. Уосермен Ф. Нейрокомпьютерная техника. Теория и практика / Ф. Уосермен. – М., 1992. – 184 с.
8. Ануфриев И. Е. MatLab 7 / И. Е. Ануфриев, А. Б. Смирнов, Е. Н. Смирнова. – СПб. : БХВ-Петербург, 2005. – 1104 с.
9. Дащенко А. Ф. MathLab в инженерных и научных расчетах / А. Ф. Дащенко, В. Х. Кириллов, Л. В. Коломиец, В. Ф. Оробей. – Одесса : Астропринт, 2003. – 214 с.
10. Гонсалес Р. Цифровая обработка изображений в среде MatLab / Р. Гонсалес, Р. Вудс, С. Эддинс. – М. : Техносфера, 2006. – 616 с.

ЛАБОРАТОРНА РОБОТА № 4
ВИКОРИСТАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Мета: вивчити принципи розв'язання оптимізаційних задач за допомогою генетичних алгоритмів, одержати навички програмування генетичних алгоритмів.

Завдання: згідно з варіантом провести моделювання еволюції форми дерева в програмі „*Gen*_11_4" при існуючій структурі хромосом; дослідити вплив параметрів генетичних алгоритмів на ефективність еволюції; створити нову структуру хромосом та провести для неї моделювання еволюції форми дерева; дослідити ефективність різних видів селекції хромосом.

Обладнання: персональний комп'ютер.

Програмне забезпечення: програма „*Gen*_11_4", середовище розробки програм *Delphi*.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Основи генетичних алгоритмів

Генетичні алгоритми (*Genetic Algorithms*) є складовою еволюційних методів [1 - 4], оскільки виникли у результаті спроб копіювання еволюції живих організмів. Генетичні алгоритми (ГА) – це процедури пошуку, засновані на механізмах природного відбору і спадкування; в ГА використовується еволюційний принцип виживання найбільш пристосованих особин. Генетичні алгоритми мають певні переваги в порівнянні з традиційними методами оптимізації, оскільки поєднують кращі властивості градієнтних методів та методів повного перебору. Порівняно з градієнтними методами ГА забезпечують кращий результат, а порівняно з методами повного перебору ГА мають значно вищу швидкодію. Тому ГА широко використовуються в системах штучного інтелекту при розв'язанні різноманітних оптимізаційних задач.

За допомогою генетичних алгоритмів задача розв'язується так. Уявімо собі штучний світ, населений кількістю N істот (особин), причому генетичний код кожної істоти – це деякий розв'язок нашої задачі. Разом усі особини утворюють популяцію P_o. Для простоти вважається, що генетичний код кожної особини (генотип) записується у вигляді однієї хромосоми (*Chromosome*) X (для людини генотип містить 46 хромосом). У ГА хромосома X особини є впорядкованою послідовністю з M генів, тобто хромосома $X = \{G_1, G_j, ..., G_M\}$ –

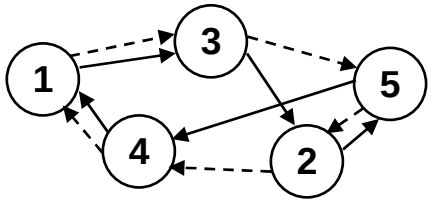
числовий вектор, який описує розв'язок задачі. Генотип усіх особин утворює популяцію хромосом $P = \{X_1, X_i, ...X_N\}$.

У класичному ГА кожний ген записується у вигляді набору R бітів, тому вся хромосома записується набором двійкових чисел (алелів) довжиною $L = M \cdot R$ (подібно до молекули ДНК). Кожен ген кодує певну властивість організму (в людини більше 60 тис. генів).

Особина вважається тим більш пристосованою, чим кращий розв'язок задачі вона забезпечує (більше значення цільової функції). В ГА цільова функція також називається функцією пристосованості або фітнес-функцією (*fitness function*) F. Тоді задача оптимізації цільової функції зводиться до пошуку найбільш пристосованої особини. У віртуальному світі ГА буде існувати N істот (особин), покоління яких змінюють одне одного. Тепер, якщо ввести в дію природний відбір і генетичне наслідування, то одержаний світ підкорятиметься законам еволюції.

Розглянемо принципи роботи генетичних алгоритмів на прикладі класичної задачі комівояжера (*TSP - travelling salesman problem*). Задача формулюється так: комівояжер повинен об'їхати M міст, побувавши в кожному один раз, і повернутися в початкову точку. Цільовою функцією є довжина шляху S, яка повинна мінімізуватися. Виявляється, що вже для $M = 31$ міст пошук оптимального шляху традиційними методами (наприклад, повного перебору) є складною задачею (потрібно перебрати $30! \approx 10^{32}$ маршрутів, що нереально). Тому це спонукало до розвитку нових методів оптимізації, зокрема генетичних алгоритмів.

У випадку задачі комівояжера кожний маршрут (фенотип) описується рядком з M десяткових чисел, тобто хромосома X особини є впорядкованою послідовністю з M генів, де на j-му місці стоїть номер j-го по порядку обходу міста. Якщо на початку розв'язування задачі комівояжера ми маємо випадковий набір маршрутів (особин), то в процесі еволюції виживуть тільки маршрути з мінімальною довжиною (кращим набором генів) (рис. 1.1).



а

X_1 {1, 3, 2, 5, 4}; S=12
X_2 {1, 3, 5, 2, 4}; S=11
...
X_N {1, 3, 5, 4, 2}; S=13

б

Рис. 1.1. Розв'язання задачі комівояжера за допомогою ГА: *а* – фенотип (маршрути); *б* – генотип (набір хромосом популяції); S – довжина шляху

При комп'ютерній імітації еволюції часто використовується стратегія елітизму, яка полягає у збереженні кращого з індивідуумів поточного покоління.

ГА відрізняються від традиційних задач оптимізації кількома моментами:

1) ГА оброблюють не значення параметрів самої задачі, а їх закодовану форму;
2) виконують пошук розв'язку виходячи не з однієї точки, а з деякої популяції;
3) використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;
4) використовують ймовірнісні, а не детерміновані правила відбору.

1.2. Класичний генетичний алгоритм

Класичний ГА складається з таких кроків (рис. 1.2):

1. Ініціалізація, або вибір початкової популяції хромосом.
2. Оцінювання пристосованості хромосом у популяції.
3. Перевірка умови закінчення алгоритму.
4. Селекція хромосом.
5. Використання генетичних операторів.
6. Формування нової популяції.
7. Вибір найкращої хромосоми.

Розглянемо ГА більш детально:

1. *Ініціалізація* (вибір початкової популяції хромосом) полягає у випадковому виборі заданої кількості N хромосом.

2. *Оцінювання пристосованості хромосом* у популяції полягає в розрахунку функції пристосованості для кожної хромосоми $F(X_i)$. Приймається, що функція пристосованості завжди набуває невід'ємні значення. Звичайно знаходиться максимум цієї функції.

3. *Перевірка умови закінчення алгоритму*. Алгоритм може бути закінчений, якщо його виконання не приводить до поліпшення отриманого результату, або через певний проміжок часу.

4. *Селекція хромосом* полягає у виборі на основі функції пристосованості тих хромосом, які будуть брати участь у створення нащадків, тобто нового покоління. Такий вибір виконується згідно з принципом природного відбору, за яким найбільші шанси на створення потомства мають хромосоми з найвищими значеннями функції пристосованості. До найбільш поширених методів селекції належить метод рулетки, ранговий та турнірний методи.

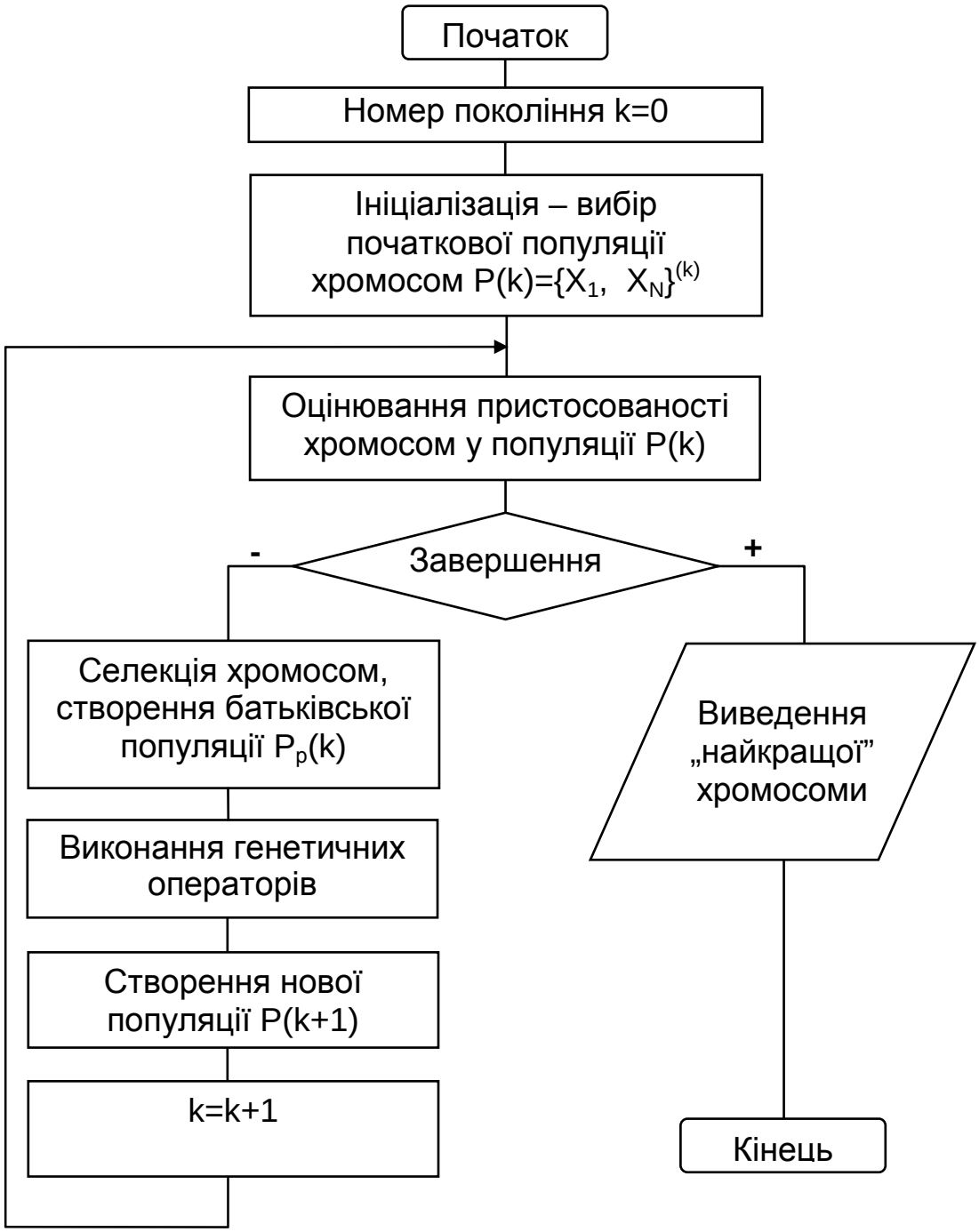


Рис. 1.2. Схема класичного генетичного алгоритму

У *методі рулетки* кожній хромосомі X_i ставиться у відповідність сектор колеса рулетки, величина якого пропорційна до функції пристосованості F даної хромосоми. Ймовірність селекції хромосоми X_i, де $i = 1...N$, дорівнює:

$$p_S(x_i) = F(x_i) / \sum_{j=1}^{N} F(x_j). \qquad (1.1)$$

Отже, ймовірність селекції в методі рулетки найбільша для хромосом із великим значенням функції пристосованості. У результаті селекції формується батьківська популяція з чисельністю N, в яку особини з великим значенням $p_S(X_i)$ можуть увійти кілька разів, а з малим значенням $p_S(X_i)$ – жодного.

При *ранговій селекції* особини популяції впорядковуються за значенням їх функції пристосованості F (за спаданням), де кожній хромосомі X ставиться у відповідність її номер i у списку (ранг). Ймовірність вибору хромосоми у батьківську популяцію в такому випадку визначається формулою

$$p_S(x_i) = \begin{cases} 1/\mu, 1 \le i \le \mu \\ 0, \mu < i \le N \end{cases}, \qquad (1.2)$$

де $\mu \le N$ – параметр методу, в найпростішому випадку $\mu = N/2$.

При *турнірному відборі* з популяції, яка містить N хромосом, у випадковий спосіб вибирається t хромосом, краща з яких (переможець) записується у масив батьківських хромосом. Розмір туру $2 \le t < N$ звичайно вибирається рівним 2 (парний турнір). Далі цей процес повторюється, поли кількість батьківських хромосом не дорівнюватиме N.

5. *Використання генетичних операторів* до відібраних у результаті селекції батьківських хромосом приводить до створення популяції нащадків. У класичному ГА використовують два основних генетичних оператори: оператор схрещування (*crossover*) і оператор мутації (*mutation*). Як і в живих організмів, імовірність мутації звичайно дуже мала ($p_м < 0.1$). У ГА мутація може виконуватися на популяції батьків перед схрещуванням або на популяції нащадків, утворених у результаті схрещування.

Оператор схрещування. На першому етапі схрещування вибираються пари хромосом із батьківської популяції. Операції схрещування полягають в обміні фрагментами ланцюжків між двома батьківськими хромосомами. Далі для кожної пари вибирається позиція гена (локус) у хромосомі, який визначає точку схрещування. Якщо хромосома містить L двійкових чисел, то точка схрещування L_K вибирається випадковим способом з інтервалу [1, L]. У результаті схрещування утворюються два нащадки:

1) нащадок, хромосома якого на позиціях від 1 до L_K складається з генів першого предка, а позиції від $L_K + 1$ до L – з генів другого предка;

2) нащадок, хромосома якого на позиціях від 1 до L_K складається з генів другого предка, а позиції від L_K + 1 до L – з генів першого предка (рис. 1.3).

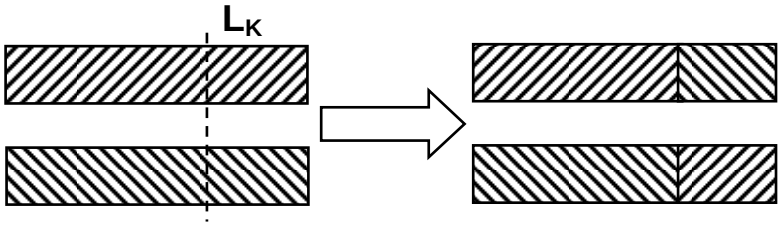


Рис. 1.3. Умовна схема схрещування

Наприклад, якщо хромосоми (1, 2, 3, 4, 5) і (0, 0, 0, 0, 0) розрізати між третім і четвертим генами й обміняти їх частини, то утворяться нащадки (1, 2, 3, 0, 0) і (0, 0, 0, 4, 5).

Оператор *мутації* з ймовірністю $p_м$ змінює значення гена на певну числову величину (амплітуду мутації A_M).

6. *Формування нової популяції*. Хромосоми, отримані у результаті дії генетичних операторів на популяцію предків (покоління *k*), утворюють нову популяцію (покоління *k*+1).

7. *Вибір найкращої хромосоми*. Найкращою вважається хромосома з максимальним значенням функції пристосованості.

1.3. Опис програми „ Gen_11_4”

Програма „*Gen*_11_4” призначена для моделювання еволюції форми дерева за допомогою генетичних алгоритмів (рис. 1.2).

На головній формі програми (рис. 1.4) моделюється процес еволюції форми дерева та встановлюються його параметри. Еволюція починається при натисненні кнопки „*Старт*”, а зупиняється при натисненні кнопки „*Стоп*”. У процесі еволюції на формі програми відображаються: номер покоління хромосом *k*, номер хромосоми *i* та номер найкращої хромосоми в популяції *i_Max*. На формі показується найкраще значення функції пристосованості для першого покоління F_0 та результуюче значення *F* (функції пристосованості або фітнес-функції).

У процесі еволюції можна змінювати режим відображення даних на формі програми за допомогою перемикачів області „*Показати*”. Значення коефіцієнтів форми дерева можна показати, якщо встановити перемикач „*Коефіцієнти*”. Увімкнений перемикач „*Еліту*” відображає найкращу хромосому в кожній популяції, а перемикач „*Еволюцію*” показує всі хромосоми. Перемикачі „*k*” та „*i*” показують стан програми для кожного покоління *k* і номера хромосоми *i*.

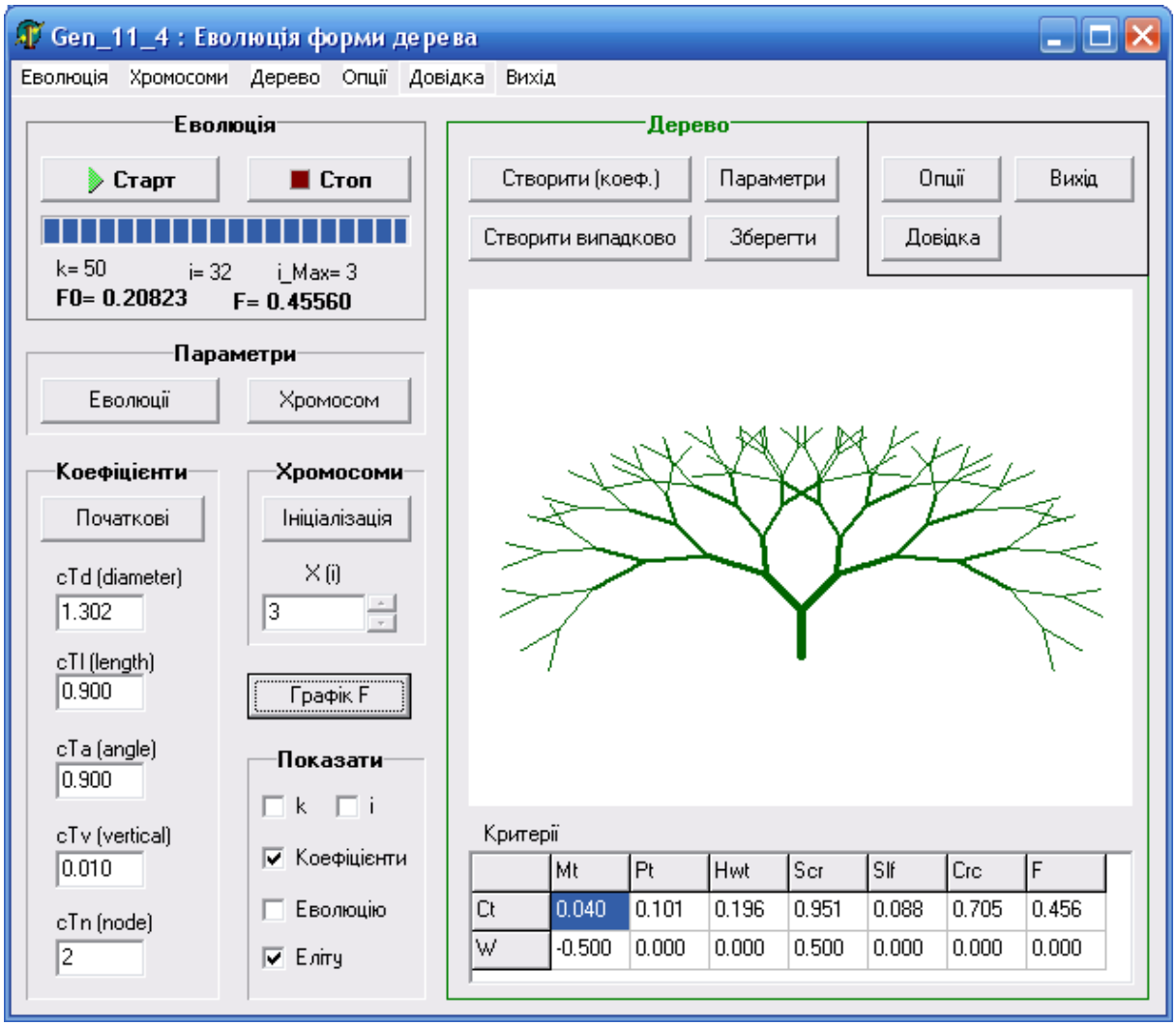


Рис. 1.4. Головна форма програми „Gen_11_4”: результат еволюції

У програмі використано модифіковані генетичні алгоритми, що як значення генів використовують дійсні числа. Для даної задачі моделювання еволюції форми дерева вибрано такий формат хромосоми: X_i (G_1, G_j,..., G_M), де $M = 5$, а гени G_j описують коефіцієнти форми дерева (*cTd*, *cTl*, *cTa*, *cTv*, *cTn*) (рис. 1.5, а). Тобто форма дерева розраховується на основі таких коефіцієнтів:

1. *cTd* – коефіцієнт діаметра гілок (*Coefficient Tree Diameter*), який дорівнює відносному збільшенню діаметра гілок на попередньому ярусі дерева відносно наступного (рис. 1.5, б, в); стовбур дерева вважається першим ярусом, а листя – останнім.
2. *cTl* – коефіцієнт довжини гілок (*Coefficient Tree Length*), який дорівнює відносному зменшенню довжини гілок на наступному ярусі дерева відносно попереднього (рис. 1.5, г, д).
3. *cTa* – коефіцієнт кута гілок (*Coefficient Tree Angle*), який дорівнює відносному зменшенню кута між розгалуженнями гілок на наступному ярусі дерева порівняно з попереднім (рис. 1.5, е, є).

4. *cTv* – коефіцієнт вертикальності гілок (*Coefficient Tree Vertical*); при значенні коефіцієнта 0.5 довжина гілки не залежить від її нахилу; при *cTv* < 0.5 довжина гілки тим більша, чим більше відхилення гілки від вертикалі; при *cTv* > 0.5 довжина гілки тим більша, чим менше відхилення гілки від вертикалі (рис. 1.5, ж,з).
5. *cTn* – коефіцієнт розгалуження (*Coefficient Tree Node* дорівнює кількості гілок, на які розгалужуються гілка попереднього рівня, коефіцієнт набуває значення 2 або 3 (рис. 1.5, а, и).

Оскільки хромосома складається з коефіцієнтів форми дерева, то формат хромосоми X_i (*cTd, cTl, cTa, cTv, cTn*). Значення коефіцієнтів форми дерева за замовчуванням встановлюються кнопкою «Початкові».

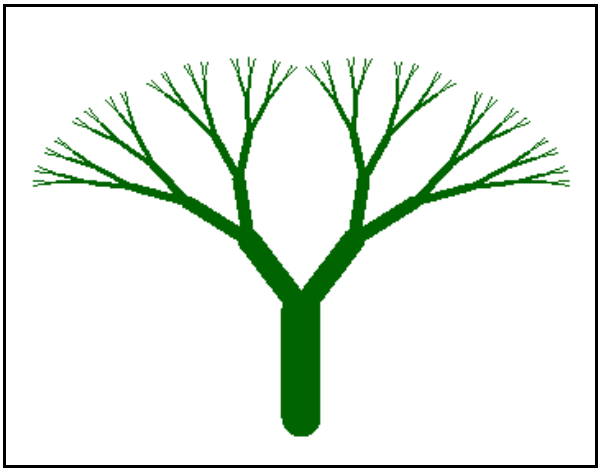
Ініціалізація значень хромосом виконується кнопкою „*Ініціалізація*”, після чого значення генів можна переглядати, встановивши номер хромосоми X_i. Ініціалізація означає створення популяції хромосом $P = \{X_1, X_i, ...X_N\}$, які описують форму дерева.

Створення і збереження дерева, перегляд його параметрів виконується кнопками в області „*Дерево*”. Кнопка „*Створити (коеф.)*” створює дерево на основі коефіцієнтів, які зчитуються з форми. Кнопка „*Створити випадково*” створює дерево на основі коефіцієнтів, які вибираються випадково з допустимого діапазону для генів хромосом. Кнопка „*Параметри*” відкриває форму з параметрами дерева, де встановлюється максимальна кількість ярусів дерева *qL* ($qL \le 8$). Кнопка „*Зберегти*” зберігає зображення дерева у файл.

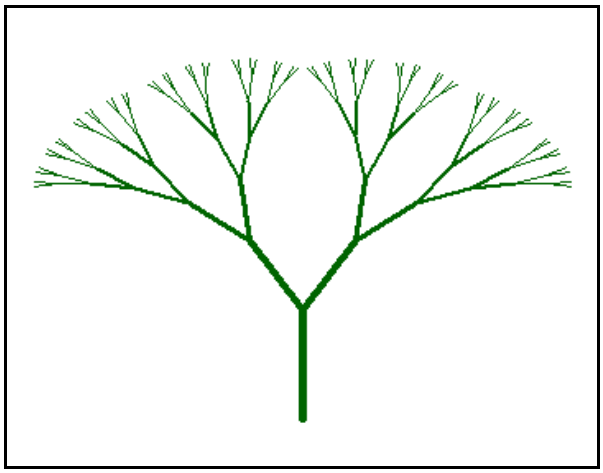
На основі форми дерева обчислюються значення 6-ти критеріїв *Ct*, які служать для розрахунку функції пристосованості *F*. При цьому кожний критерій має свою вагу *W*, яку можна встановлювати в таблиці на формі (рис. 1.4). Характеристики форми дерева описуються такими критеріями:

1. *Mt* – маса дерева (*Mass of tree*).
2. *Pt* – максимальний тиск на гілки дерева (*Pressure of tree /Max/*).
3. Hwt – відношення висоти дерева до його ширини (He*ight/ width of tree*).
4. *Scr* – площа поверхні крони дерева (*Surface of crown of tree*).
5. *Slf* – площа освітленої поверхні листя, тобто гілок останнього ярусу (*Surface of leaf of tree*).
6. *Crc* – відношення центру ваги крони до висоти дерева (*Crown center of tree*).

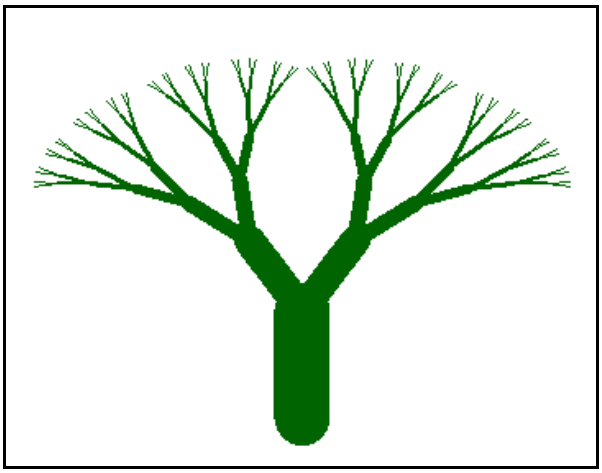
На формі „*Параметри еволюції*” для кожного гена встановлюється нижня та верхня межі (рис. 1.6).



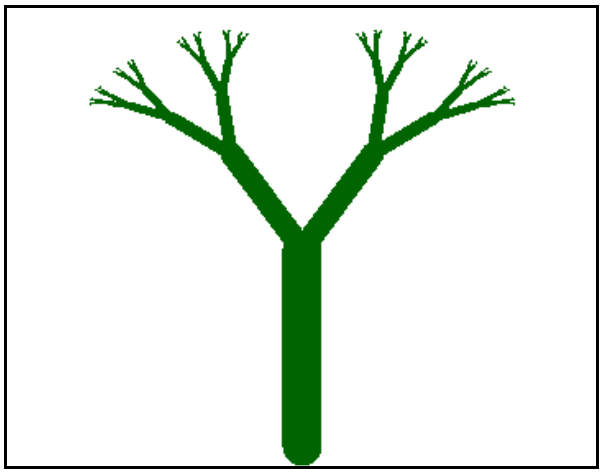
а) X_1 (1.70, 0.75, 0.75, 0.50, 2) – коефіцієнти форми дерева за замовчуванням



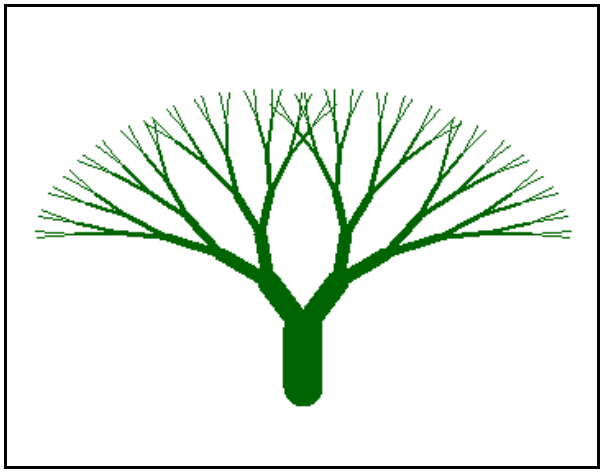
б) X_2 (1.30, 0.75, 0.75, 0.50, 2)



в) X_3 (1.80, 0.75, 0.75, 0.50, 2)

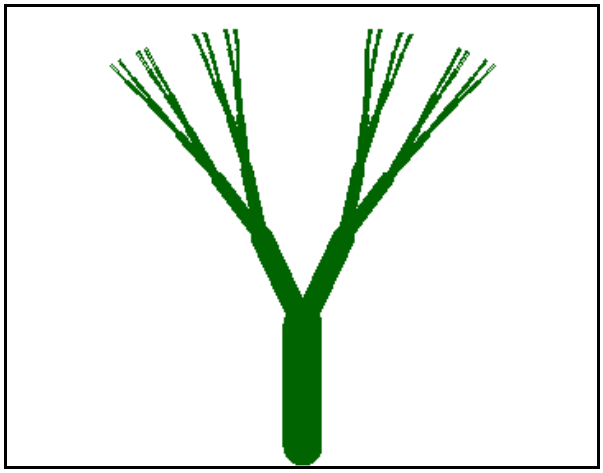


г) X_4 (1.70, 0.55, 0.75, 0.50, 2)

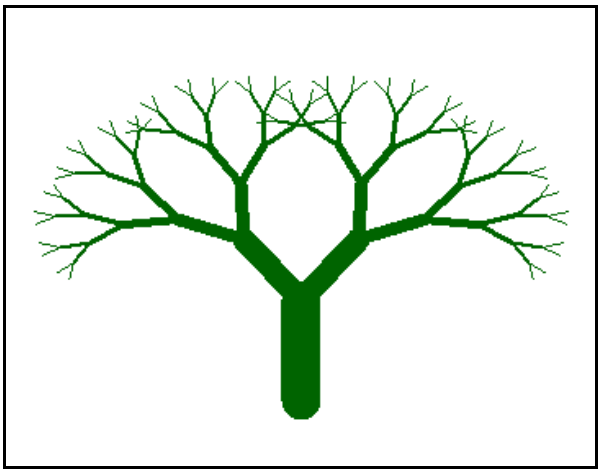


д) X_5 (1.70, 0.90, 0.75, 0.50, 2)

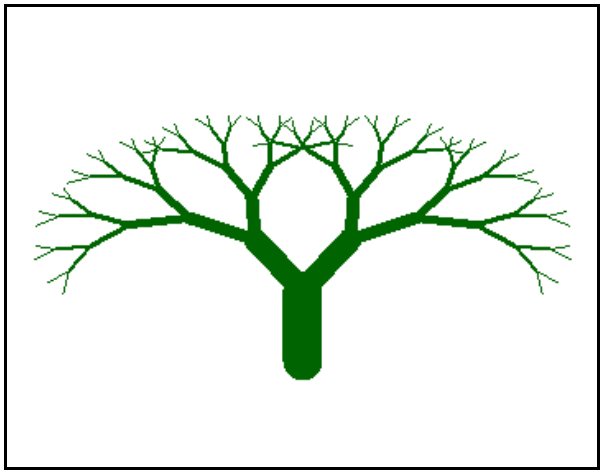
Рис. 1.5. Залежність форми дерева (фенотипу) від значень генів хромосоми (генотипу) X_i (cTd, cTl, cTa, cTv, cTn)



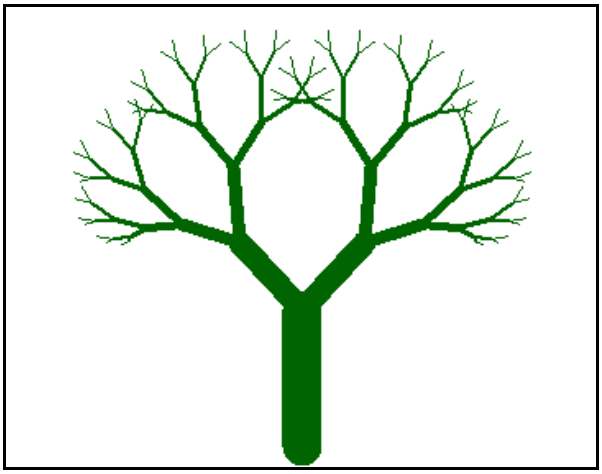
е) X_6 (1.70, 0.75, 0.50, 0.50, 2)



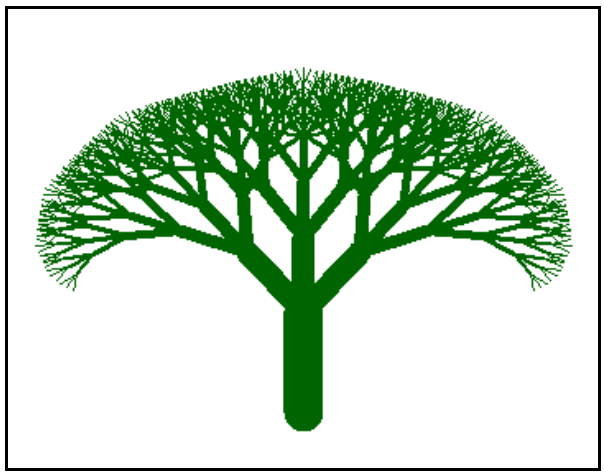
є) X_7 (1.70, 0.75, 0.90, 0.50, 2)



ж) X_8 (1.70, 0.75, 0.90, 0.01, 2)



з) X_9 (1.70, 0.75, 0.90, 0.99, 2)



и) X_{10} (1.70, 0.75, 0.90, 0.50, 3)

Продовження рис. 1.5

З форми „*Параметри хромосом*” зчитуються параметри ГА: кількість поколінь Q_k, кількість хромосом N ($N \le 256$), максимальне значення фітнес-функції *F_Max*, ймовірність мутації P_M, амплітуда мутації A_M (рис. 1.7, рис. 1.8). Значення генів хромосоми відображаються в табличному вигляді, а також як зображення *Image_X*.

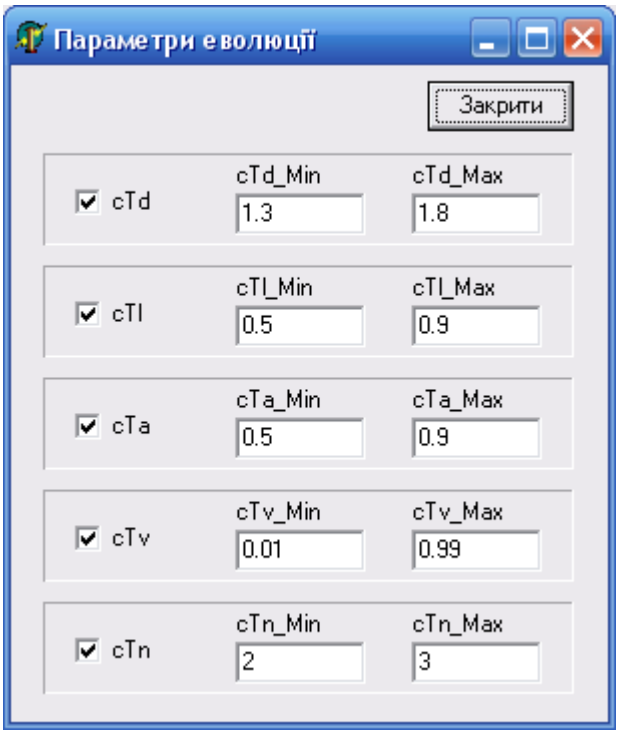


Рис. 1.6. Форма „Параметри еволюції”: мінімальні та максимальні значення генів

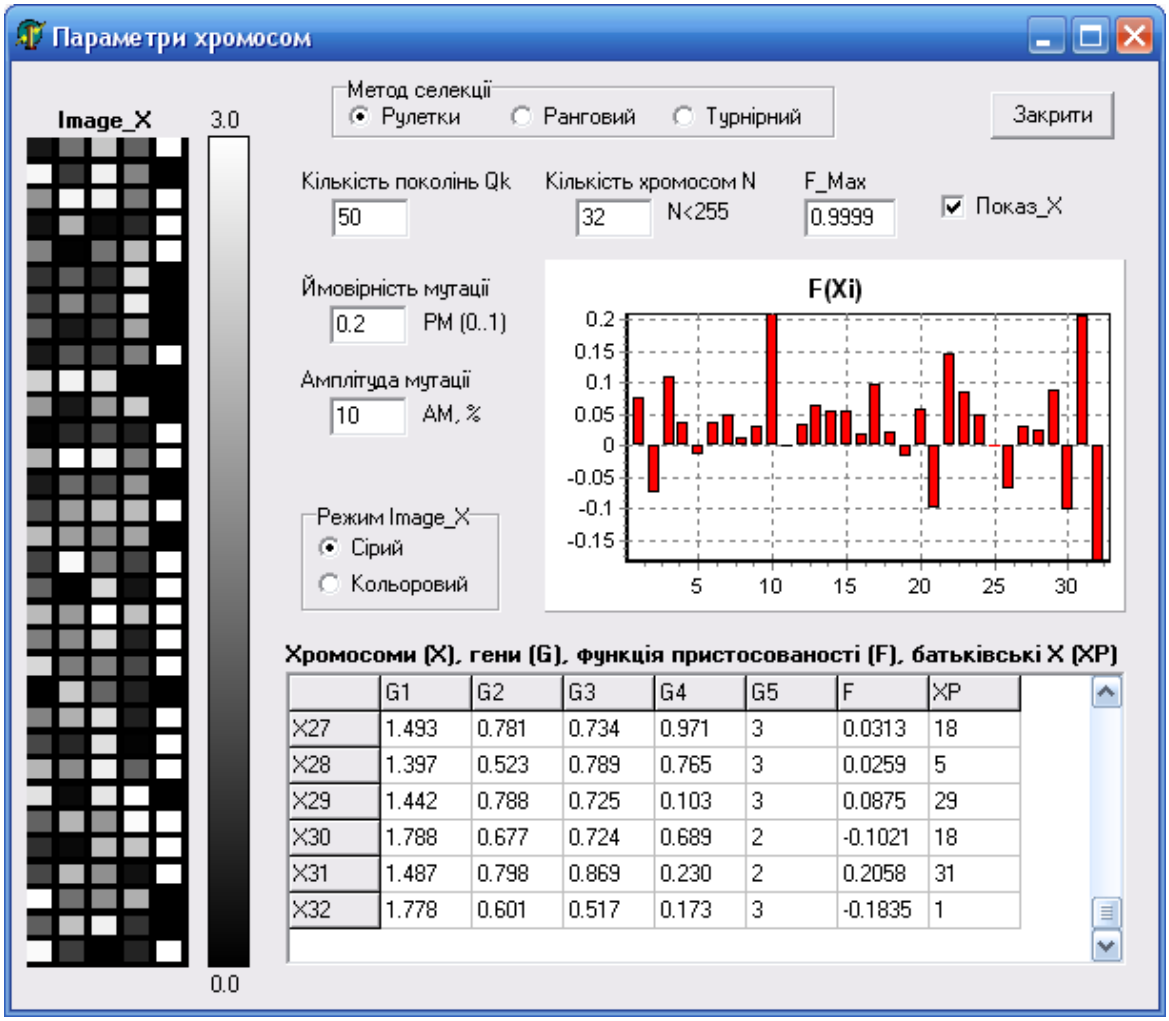


Рис. 1.7. Форма „Параметри хромосом”: стан хромосом після ініціалізації

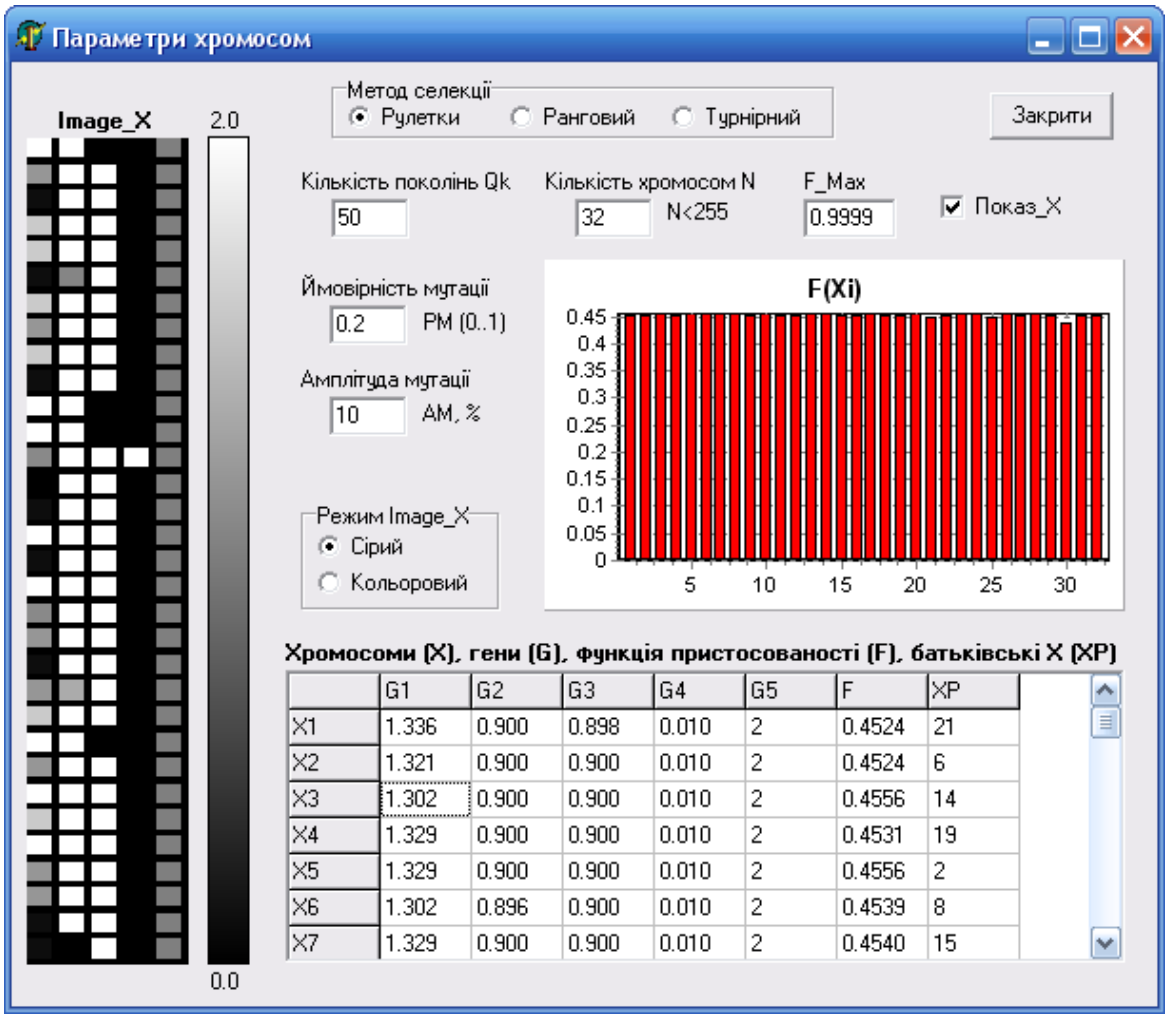


Рис. 1.8. Форма „Параметри хромосом”: стан хромосом після завершення еволюції

Після встановлення ваг *W* критеріїв, які описують форму дерева, розпочинається процес еволюції, тобто оптимізації форми дерева згідно з вибраними критеріями *Ct*. У процесі еволюції відбираються кращі форми дерев, тому функція пристосованості хромосом *F* зростає протягом певної кількості поколінь *k* (рис. 1.9). У результаті отримується хромосома з максимальним значенням *F*, яка відповідає формі дерева, близькій до оптимальної (рис. 1.4, рис. 1.8).

На формі „*Опції*” встановлюється мова інтерфейсу, а також час затримки *Time_Sleep_Tree* (в мс) при відображенні дерева на формі в процесі еволюції. За рахунок часу затримки можна візуально спостерігати еволюцію форми дерева.

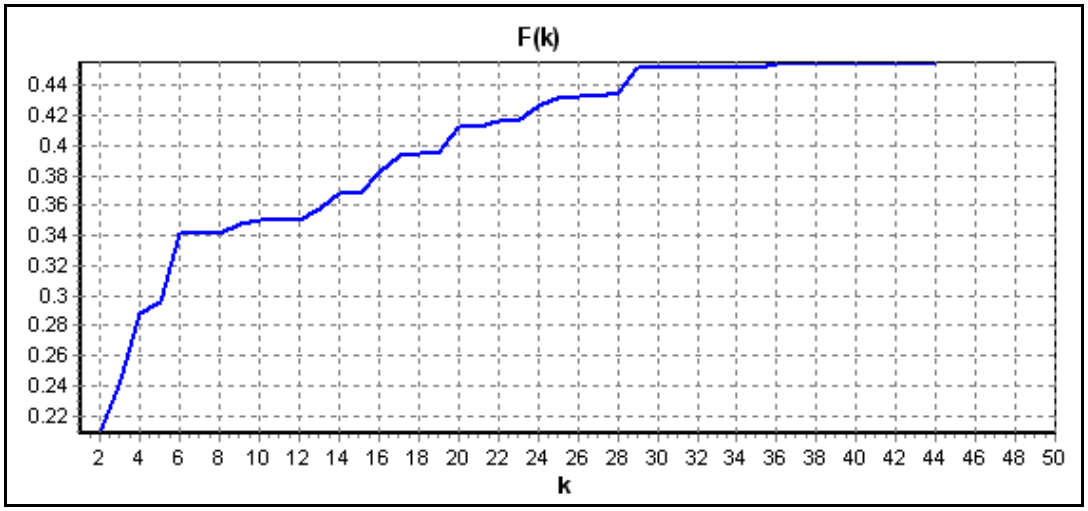


Рис. 1.9. Фрагмент форми „Графік F(k)”: функція пристосованості F зростає

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для варіантів V 1...V 100

10. Ознайомитися з теоретичними відомостями та кодом програми „*Gen*_11_4”. За допомогою програми „*Gen*_11_4” дослідити вплив коефіцієнтів на форму дерева (рис. 1.4, рис. 1.5).
11. Провести моделювання еволюції форми дерева в програмі „*Gen*_11_4” при існуючій структурі хромосом і для ваг *W* критеріїв *Ct* форми дерева згідно з варіантом (табл. 2.1). Моделювання еволюції виконати при кількості хромосом $N = V \cdot 2 + 16$; імовірності мутації $P_M = V \cdot 0.005 + 0.05$, амплітуді мутації $A_M = V \cdot 0.005 + 0.05$, кількості поколінь $Q_k = 100$. Моделювання форми дерева виконати 5 разів, два кращі результати еволюції (зображення головної форми програми, форми „*Параметри хромосом*”, форми „*Графік F(k)*”) додати до звіту. Провести аналіз отриманої форми дерев та їх генотипу.
12. Дослідити вплив параметрів генетичних алгоритмів на ефективність еволюції: повторити п.2 при кількості хромосом $N = V \cdot 2 + 26$; імовірності мутації $P_M = V \cdot 0.005 + 0.1$.

Таблиця 2.1

Ваги *W* критеріїв *Ct* форми дерева (згідно з номером варіанта *V*)

V	*Mt*	*Pt*	Hwt	*Scr*	*Slf*	*Crc*
0	0.1	0	0.65	0.7	-0.13	0
1	-0.5	0	0	0.5	0	0
2	-0.5	0	0.5	0.5	0	0
3	-0.4	0	0.5	0.5	0	0
4	-0.3	0	0.5	0.5	0	0
5	-0.2	0	0.5	0.5	0	0
6	-0.1	0	0.5	0.5	0	0
7	-0.01	0	0.5	0.5	0	0
8	0	0	0.5	0.5	0	0
9	0.01	0	0.5	0.5	0	0
10	0.05	0	0.5	0.5	0	0
11	0.1	0	0.5	0.5	0	0
12	0.15	0	0.5	0.5	0	0
13	0.2	0	0.5	0.5	0	0
14	0.3	0	0.5	0.5	0	0
15	0.4	0	0.5	0.5	0	0
16	0.5	0	0.5	0.5	0	0
17	0.9	0	0.5	0.5	0	0
18	-0.5	0	0.6	0.5	0	0
19	0.1	0	0.6	0.5	0	0
20	0.5	0	0.6	0.5	0	0
21	0.5	0	0.6	0.7	0	0
22	0.1	0	0.7	0.5	0	0
23	0.1	0	0.6	0.7	-0.1	0
24	0.1	0	0.6	0.7	-0.3	0
25	0.2	0	0.6	0.75	-0.3	0
26	0.5	0	0.7	0.75	-0.3	0
27	0.1	0	0.3	0	0.3	0
28	0.1	0	-0.1	0	0.3	0
29	-0.1	0	-0.5	0	0.5	0
30	0	-0.5	-0.5	0	0.5	0
31	0	-0.8	-0.5	0	0.5	0
32	0	-0.5	-0.1	0	0.5	0
33	0	-0.5	-0.05	0	0.5	0

Продовження таблиці 2.1

V	*Mt*	*Pt*	Hwt	*Scr*	*Slf*	*Crc*
34	0	-0.5	-0.02	0	0.5	0
35	0	-0.5	-0.01	0	0.5	0
36	0	-0.5	-0.001	0	0.5	0
37	0	-0.5	0	0	0.5	0
38	0	-0.5	0.001	0	0.5	0
39	0	-0.5	0.005	0	0.5	0
40	0	-0.5	0.01	0	0.5	0
41	0	-0.5	0.1	0	0.5	0
42	0	-0.5	0.5	0	0.5	0
43	0	-0.5	-0.01	0	-0.1	0
44	-0.1	-0.5	-0.02	0	-0.1	0
45	0.1	-0.5	-0.02	0	-0.1	0
46	0.2	-0.5	-0.02	0	-0.1	0
47	0.3	-0.5	-0.02	0	-0.1	0
48	0.4	-0.5	-0.02	0	-0.1	0
49	0.5	-0.5	-0.02	0	-0.1	0
50	-0.01	-0.5	-0.02	0	-0.1	0.5
51	-0.01	-0.5	-0.02	0	-0.1	0.1
52	-0.01	-0.5	-0.02	0	-0.1	0.05
53	-0.01	-0.5	-0.02	0	-0.1	0.01
54	-0.01	-0.5	-0.02	0	-0.1	0.001
55	0.01	-0.5	-0.02	0	-0.1	-0.001
56	-0.01	-0.5	-0.02	0	-0.1	-0.01
57	-0.01	-0.5	-0.02	0	-0.1	-0.1
58	-0.01	-0.5	-0.02	0	-0.1	-0.15
59	0.01	-0.5	-0.02	0	-0.1	-0.2
60	-0.01	-0.5	-0.02	0	-0.1	-0.25
61	0.01	-0.5	-0.02	0	-0.1	-0.5
62	0.01	-0.5	-0.02	0.1	-0.1	-0.5
63	0.01	-0.5	-0.02	0.05	-0.1	-0.5
64	0.01	-0.5	-0.02	-0.05	-0.1	-0.5
65	0.01	-0.5	-0.02	-0.1	-0.1	-0.5
66	0.01	-0.5	-0.02	-0.2	-0.1	-0.5
67	0.01	-0.5	-0.02	-0.3	-0.1	-0.5

Продовження таблиці 2.1

V	*Mt*	*Pt*	Hwt	*Scr*	*Slf*	*Crc*
68	0.01	-0.5	-0.02	-0.4	-0.1	-0.5
69	0.01	-0.5	-0.02	-0.5	-0.1	-0.5
70	0.01	-0.5	-0.5	-0.5	-0.1	-0.5
71	0.01	-0.5	-0.5	-0.4	-0.1	-0.5
72	0.01	-0.5	-0.5	-0.3	-0.1	-0.5
73	0.01	-0.5	-0.5	-0.2	-0.1	-0.5
74	0.01	-0.5	-0.5	-0.1	-0.1	-0.5
75	0.01	-0.5	-0.5	0	-0.1	-0.5
76	-0.5	0	0.2	0.5	0	0
77	-0.45	0	0.5	0.5	0	0
78	-0.35	0	0.5	0.5	0	0
79	-0.25	0	0.5	0.5	0	0
80	-0.15	0	0.5	0.5	0	0
81	-0.05	0	0.5	0.5	0	0
82	-0.005	0	0.5	0.5	0	0
83	0.005	0	0.5	0.5	0	0
84	0.02	0	0.5	0.5	0	0
85	0.075	0	0.5	0.5	0	0
86	0.12	0	0.5	0.5	0	0
87	0.175	0	0.5	0.5	0	0
88	0.25	0	0.5	0.5	0	0
89	0.35	0	0.5	0.5	0	0
90	0.45	0	0.5	0.5	0	0
91	0.75	0	0.5	0.5	0	0
92	-0.4	0	0.5	0.5	0	0
93	-0.2	0	0.6	0.5	0	0
94	0.15	0	0.65	0.5	0	0
95	0.35	0	0.65	0.5	0	0
96	0.35	0	0.65	0.75	0	0
97	0.15	0	0.7	0.5	0	0
98	0.15	0	0.65	0.7	-0.1	0
99	0.15	0	0.6	0.7	-0.2	0
100	0.25	0	0.65	0.75	-0.3	0

13. Створити нову структуру хромосом. Кількість генів хромосоми потрібно збільшити до 6, призначення гену G_6 вибрати згідно з варіантом (табл. 2.2). Нову кількість генів M встановити в процедурі „*FormCreate*" головної форми програми. Для створення нової структури хромосом у коді програми потрібно зробити такі зміни:
    - У процедурі ініціалізації хромосом „*pG1_Init_X_chromosom*" для встановлення початкових значень генів потрібно дописати програмний код, який розраховує нове значення гена G_6 і записує його в масив *mXG*[*i*,*j*] (*i* – номер хромосоми, *j* – номер гена). Значення нового гена обчислюється аналогічно до інших генів через мінімальне і максимальне значення (використати мінімальні та максимальні значення інших генів, записані в масивах mG_Min[j] та mG_Max[j] відповідно).
    - В кінці процедури „*p_Tree_Level_Init*" потрібно дописати код, який обчислює параметр ярусу L дерева через значення гена *mXG*[*i*,*j*] та параметр іншого ярусу. Наприклад, якщо ген G_6 описує діаметр гілок ярусу $L = 3$ і за умовою параметр ярусу L обчислюється через параметр ярусу $L+1$, то

        mLd[3]:=mLd[3+1]* mXG[i,6].

    Лістинг змінених процедур „*pG1_Init_X_chromosom*" та „*p_Tree_Level_Init*" з коментарями додати до звіту. Повторити п.2 для нової структури хромосом.

14. *Додаткове завдання*. В програмі „*Gen*_11_4" створити нові процедури селекції хромосом: для непарних варіантів селекцію виконувати ранговим методом, для непарних – турнірним. Повторити п.2 для нового методу селекції хромосом. Лістинг створеної процедури з коментарями додати до звіту.

Таблиця 2.2

Призначення нового гена G_6, який описує параметри ярусу *L* (згідно з номером варіанта *V*)

V	Призначення гена G_6	*V*	Призначення гена G_6
Кількість ярусів *qL* = 7, параметр ярусу L обчислюється через параметр ярусу L+1			
0	Діаметр гілок ярусу 1	3	Діаметр гілок ярусу 4
1	Діаметр гілок ярусу 2	4	Діаметр гілок ярусу 5
2	Діаметр гілок ярусу 3	5	Діаметр гілок ярусу 6
Кількість ярусів *qL* = 7, параметр ярусу L обчислюється через параметр ярусу L-1			
6	Довжина гілок ярусу 2	18	Коефіцієнт вертикальності ярусу 2
7	Довжина гілок ярусу 3	19	Коефіцієнт вертикальності ярусу 3
8	Довжина гілок ярусу 4	20	Коефіцієнт вертикальності ярусу 4
9	Довжина гілок ярусу 5	21	Коефіцієнт вертикальності ярусу 5
10	Довжина гілок ярусу 6	22	Коефіцієнт вертикальності ярусу 6
11	Довжина гілок ярусу 7	23	Коефіцієнт вертикальності ярусу 7
12	Кутовий діапазон ярусу 2	24	Кількість розгалужень ярусу 2
13	Кутовий діапазон ярусу 3	25	Кількість розгалужень ярусу 3
14	Кутовий діапазон ярусу 4	26	Кількість розгалужень ярусу 4
15	Кутовий діапазон ярусу 5	27	Кількість розгалужень ярусу 5
16	Кутовий діапазон ярусу 6	28	Кількість розгалужень ярусу 6
17	Кутовий діапазон ярусу 7	29	Кількість розгалужень ярусу 7
Кількість ярусів *qL* = 7, параметр ярусу L обчислюється через параметр ярусу 1			
30	Довжина гілок ярусу 2	39	Кутовий діапазон ярусу 5
31	Довжина гілок ярусу 3	40	Кутовий діапазон ярусу 6
32	Довжина гілок ярусу 4	41	Кутовий діапазон ярусу 7
33	Довжина гілок ярусу 5	42	Коефіцієнт вертикальності ярусу 2
34	Довжина гілок ярусу 6	43	Коефіцієнт вертикальності ярусу 3
35	Довжина гілок ярусу 7	44	Коефіцієнт вертикальності ярусу 4
36	Кутовий діапазон ярусу 2	45	Коефіцієнт вертикальності ярусу 5
37	Кутовий діапазон ярусу 3	46	Коефіцієнт вертикальності ярусу 6
38	Кутовий діапазон ярусу 4	47	Коефіцієнт вертикальності ярусу 7

Продовження таблиці 2.2

V	Призначення гена G_6	V	Призначення гена G_6
Кількість ярусів qL = 8, параметр ярусу L обчислюється через параметр ярусу L+1			
48	Діаметр гілок ярусу 1	52	Діаметр гілок ярусу 5
49	Діаметр гілок ярусу 2	53	Діаметр гілок ярусу 6
50	Діаметр гілок ярусу 3	54	Діаметр гілок ярусу 7
51	Діаметр гілок ярусу 4		
Кількість ярусів qL = 8, параметр ярусу L обчислюється через параметр ярусу L-1			
55	Довжина гілок ярусу 2	69	Коефіцієнт вертикальності ярусу 2
56	Довжина гілок ярусу 3	70	Коефіцієнт вертикальності ярусу 3
57	Довжина гілок ярусу 4	71	Коефіцієнт вертикальності ярусу 4
58	Довжина гілок ярусу 5	72	Коефіцієнт вертикальності ярусу 5
59	Довжина гілок ярусу 6	73	Коефіцієнт вертикальності ярусу 6
60	Довжина гілок ярусу 7	74	Коефіцієнт вертикальності ярусу 7
61	Довжина гілок ярусу 8	75	Кількість розгалужень ярусу 2
62	Кутовий діапазон ярусу 2	76	Кількість розгалужень ярусу 3
63	Кутовий діапазон ярусу 3	77	Кількість розгалужень ярусу 4
64	Кутовий діапазон ярусу 4	78	Кількість розгалужень ярусу 5
65	Кутовий діапазон ярусу 5	79	Кількість розгалужень ярусу 6
66	Кутовий діапазон ярусу 6	80	Кількість розгалужень ярусу 7
67	Кутовий діапазон ярусу 7	81	Кількість розгалужень ярусу 8
68	Кутовий діапазон ярусу 8		
Кількість ярусів qL = 8, параметр ярусу L обчислюється через параметр ярусу 1			
82	Довжина гілок ярусу 2	92	Кутовий діапазон ярусу 5
83	Довжина гілок ярусу 3	93	Кутовий діапазон ярусу 6
84	Довжина гілок ярусу 4	94	Кутовий діапазон ярусу 7
85	Довжина гілок ярусу 5	95	Кутовий діапазон ярусу 8
86	Довжина гілок ярусу 6	96	Коефіцієнт вертикальності ярусу 2
87	Довжина гілок ярусу 7	97	Коефіцієнт вертикальності ярусу 3
88	Довжина гілок ярусу 8	98	Коефіцієнт вертикальності ярусу 4
89	Кутовий діапазон ярусу 2	99	Коефіцієнт вертикальності ярусу 5
90	Кутовий діапазон ярусу 3	100	Коефіцієнт вертикальності ярусу 8
91	Кутовий діапазон ярусу 4		

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Що таке генетичні алгоритми?
2. Опишіть походження генетичних алгоритмів.
3. Яка область використання генетичних алгоритмів?
4. Як кодуються умови задачі в генетичних алгоритмах?
5. Вкажіть значення термінів „популяція”, „хромосома”, „ген” у генетичних алгоритмах.
6. У чому різниця між генотипом і фенотипом?
7. Опишіть класичний генетичний алгоритм.
8. Назвіть види селекції хромосом.
9. У чому полягає операція схрещування?
10. Розкрийте сутність операції мутації.
11. Як у генетичних алгоритмах формується нове покоління?
12. Які переваги та недоліки генетичних алгоритмів?
13. Яка структура хромосом використовується в програмі „Gen_11_4”?
14. Як виконується ініціалізація і селекція хромосом у програмі „Gen_11_4”?
15. Як виконується схрещування і мутація хромосом у програмі „Gen_11_4”?
16. Опишіть, як у програмі „Gen_11_4” форма дерева визначається значеннями генів хромосоми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вороновский Т. К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Т. К. Вороновский, К. В. Махотило. – Харьков : Основа, 1997. – 112 с.
2. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : КМ Академія, 2002. – 336 с.
3. Люгер Дж. Ф. Искусственный интеллект: Стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2003. – 864 с.
4. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М. : Горячая линия – Телеком, 2004. – 452 с.

ЛАБОРАТОРНА РОБОТА № 5
ПРОЕКТУВАННЯ ЕКСПЕРТНОЇ СИСТЕМИ

Мета: набути навички проектування експертної системи, зокрема визначення складу знань та засвоїти технології здобуття експертних знань, що стосуються визначеної предметної області.

Обладнання і програмне забезпечення: персональний комп'ютер, середовище програмування Delphi 6.0.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Експертні системи та бази знань

Мета побудови *експертних систем* (ЕС) полягає в розробці програм, що частково чи повністю замінюють експерта-людину при розв'язанні задач у визначеній предметній області. Як самостійний напрямок ЕС сформувалися в середині 60-х років XX ст.

Інформація ЕС зберігається у *базах знань* (БЗ), які являють собою формалізовані емпіричні знання експертів у якійсь вузькій предметній області. Здобуття знань – це передача досвіду розв'язання задачі від джерела знань і перетворення його до вигляду, який дозволяє використовувати ці знання в ЕС. Проблеми здобуття знань розв'язують експерти зі знань.

При створенні БЗ перш за все необхідно визначити склад знань, тобто визначити, „Що представляти" в експертній системі. Далі необхідно визначити, „Як представляти" знання. Ці дві проблеми не є незалежними. Обраний спосіб представлення знань може виявитися непридатним або неефективним для вираження знань, що стосуються предметної області задачі.

При представленні знань вирішуються такі питання:

- визначення складу знань, що представляються;
- організація знань;
- представлення знань.

Склад знань ЕС визначається такими факторами:

- проблемним середовищем;
- архітектурою експертної системи;
- потребами й цілями користувачів;
- мовою спілкування.

1.2. Визначення складу знань та їх розподіл

Для функціонування ЕС необхідні:

- знання про процес розв'язання задачі (тобто керуючі знання);
- знання про мову спілкування і способи організації діалогу;
- знання про способи представлення і модифікації цих знань;
- підтримуючі структурні і керуючі знання.

Усі знання доцільно поділяти на такі, що *інтерпретуються*, та такі, що *не інтерпретуються*. До першого типу належать знання, структуру і зміст яких „знає" ЕС. Усі інші знання відносять до другого типу, якщо компонент не усвідомлює цих знань. Знання, що не інтерпретуються, поділяються на допоміжні й підтримуючі.

Допоміжні знання зберігають інформацію про лексику і граматику мови спілкування, інформацію про структуру діалогу, вони опрацьовуються лінгвістичною компонентою системи, але хід цього опрацювання ЕС не «усвідомлює».

Підтримуючі знання виконують роль описів (обґрунтувань) інтерпретованих знань і дій системи. Підтримуючі знання розподіляються на *технологічні* і *семантичні*.

Технологічні підтримуючі знання містять відомості про час створення описуваних ними знань, про автора знань і т.п. *Семантичні* підтримуючі знання містять значущий опис знань. Вони містять інформацію про причини введення знань, про призначення знань, описують спосіб використання знань і одержуваний ефект. Підтримуючі знання мають описовий характер.

Знання, що інтерпретуються, можна розподілити на предметні знання, керуючі знання та знання про представлення.

Знання про представлення містять інформацію про те, як (у яких структурах) у системі представлені інтерпретовані знання.

Предметні знання містять дані про предметну область і способи перетворення цих даних при розв'язанні поставлених задач. У предметних знаннях можна виділити описувачі та власне предметні знання.

Описувачі містять визначену інформацію про предметні знання, таку як коефіцієнт визначеності правил і даних, міри важливості і складності. *Власне предметні знання* розподіляються на *факти* і *твердження*, що виконуються.

Факти визначають можливі значення сутностей і характеристик предметної області. *Твердження*, що виконуються, містять інформацію про те, як можна змінювати опис предметної області в ході розв'язання задач.

Керуючі знання можна розподілити на *фокусуючі* й *вирішуючі*. *Фокусуючі* знання описують, які знання варто використовувати в тій чи іншій ситуації. Вони містять відомості про найбільш перспективні об'єкти чи правила, які доцільно використовувати при перевірці відповідних гіпотез. У першому випадку увага фокусується на елементах робочої пам'яті, у другому – на правилах бази знань.

Вирішуючі знання містять інформацію, що використовується для вибору способу інтерпретації знань, що підходить до поточної ситуації. Ці знання застосовуються для вибору стратегій чи евристик, найбільш ефективних для розв'язання даної задачі.

Якісні і кількісні характеристики ЕС можуть бути значно поліпшені за рахунок використання *метазнань*, тобто знань про знання. Метазнання не представляють деяку єдину сутність, вони можуть застосовуватися для досягнення різних цілей:

1) використовуються для обґрунтування доцільності застосування правил з області експертизи;
2) для виявлення синтаксичних і семантичних помилок у предметних правилах;
3) дозволяють системі адаптуватися до оточення шляхом перебудови предметних правил;
4) дозволяють явно вказати можливості й обмеження системи.

Залежність складу знань від вимог користувача проявляється в такому:

- з якими даними і які задачі хоче розв'язувати користувач;
- які способи і методи розв'язання кращі;
- при яких обмеженнях на кількість результатів і способи їх одержання повинна бути розв'язана задача;
- які вимоги до мови спілкування й організації діалогу.

1.3. Побудова моделі предметної області

Існують два підходи до процесу побудови моделі предметної області:

1) *ознаковий (атрибутивний*) підхід передбачає наявність отриманої від експертів інформації у вигляді трійок "об'єкт – атрибут – значення атрибута", а також наявність навчальної інформації. Цей підхід розвивається в рамках напрямку, що одержав назву формування знань чи "машинного навчання" (mashine learning).

2) *структурний (когнітивний*) підхід здійснюється шляхом виділення елементів предметної області, їх взаємозв'язків і семантичних відношень.

Для *атрибутивного* підходу характерна наявність найбільш повної інформації про предметну область – про об’єкти, їх атрибути і про значення атрибутів. Окрім того, істотним моментом є використання додаткової навчальної інформації, що задається групуванням об’єктів у класи за тим чи іншим змістовим критерієм.

Трійки „об’єкт – атрибут – значення атрибута” можуть бути отримані за допомогою так званого методу *рекласифікації*, що базується на припущенні, що задача є об’єктно-орієнтованою і об’єкти задачі добре відомі експерту. Ідея методу полягає в тому, що конструюються правила (комбінації значень атрибутів), які дозволяють відрізнити один об’єкт від іншого. При наявності навчальної інформації для формування моделі предметної області можна використовувати весь арсенал методів, що розвиваються в рамках задачі розпізнавання образів.

Структурний підхід до побудови моделі предметної області передбачає виділення наступних когнітивних елементів знань:

1) поняття;
2) взаємозв’язки;
3) метапоняття;
4) семантичні відношення.

1.4. Методи побудови системи понять предметної області

Поняття предметної області повинні утворювати систему, що має такі властивості: унікальність (відсутність дублювання); повнота (досить повний опис різних процесів, фактів, явищ і т.д. предметної області); вірогідність (валідність – відповідність виділених одиниць значущої інформації їхнім реальним найменуванням) і несуперечність (відсутність омонімії).

Для побудови системи понять предметної області можуть використовуватися нижчевказані методи.

1. "*Метод локального представлення*". При його використанні експерта просять розбити задачу на підзадачі для перерахування цільових станів і опису загальних категорій цілі. Далі для кожного розбиття (локального представлення) експерт формулює інформаційні факти і дає їм чітке найменування (назву). Вважається, що для успішного розв’язання задачі побудови моделі предметної області кількість таких інформаційних фактів у кожному локальному представленні повинна приблизно дорівнювати семи.

2. "*Метод обчислення коефіцієнта використання*" базується на такій гіпотезі: елемент даних (чи інформаційний факт) може бути поняттям, якщо:

- він використовується у великій кількості підзадач;
- використовується з великою кількістю інших елементів даних;
- рідко використовується разом з іншими елементами даних у порівнянні із загальною кількістю його використання в усіх підзадачах (це і є коефіцієнт використання).

3. "*Метод формування переліку понять*" полягає в тому, що експертам (бажано, щоб їх було більше двох) дається завдання скласти список понять, що належать до досліджуваної предметної області. Поняття, виділені всіма експертами, включаються в систему понять, інші підлягають обговоренню.

4. "*Рольовий метод*" полягає в тому, що експерту дається завдання навчити інженера по знаннях розв'язанню деяких задач предметної області. Таким чином, експерт відіграє роль учителя, а інженер по знаннях – роль учня. Процес навчання записується (наприклад, на магнітофон). Потім третій учасник прослуховує запис і виписує на папері всі поняття, вжиті вчителем чи учнем.

5. "*Метод складання змісту підручника*": експерту пропонується представити ситуацію, в якій його попросили написати підручник. Необхідно скласти на папері перелік передбачуваних розділів, параграфів, пунктів і підпунктів книги.

6. "*Текстологічний метод*" формування системи понять полягає в тому, що експерту дається завдання виписати з посібників (книг за фахом) деякі елементи, які являють собою одиниці значущої інформації.

Після закінчення процесу виділення понять необхідно встановити семантичну близькість між окремими поняттями. Для цього використовується група методів встановлення взаємозв'язків. В основі встановлення взаємозв'язків лежить психологічний ефект "вільних асоціацій", а також фундаментальна категорія близькості об'єктів чи концептів.

Ефект *вільних асоціацій* полягає в наступному. Піддослідного просять відповідати на задане слово першим словом, що спало на думку. Як правило, реакція більшості випробуваних (якщо слова не були занадто незвичайними) проявляється однаково. Кількість переходів у ланцюжку може служити мірою "значущої відстані" між двома поняттями. Численні досліди підтверджують гіпотезу, що для двох будь-яких слів (понять) існує асоціативний ланцюжок, що складається не більш ніж із семи слів.

Існують такі методи встановлення взаємозв’язків між поняттями предметної області:

1. "*Метод вільних асоціацій*" базується на психологічному ефекті, описаному вище. Експерту пред’являється поняття з проханням якнайшвидше назвати перше поняття, що спало на думку, зі сформованої раніше системи понять. Далі проводиться аналіз отриманої інформації.

2. "*Метод сортування карток*". У ньому вихідним матеріалом служать виписані на картки поняття. Застосовуються два варіанти методу. В першому експерту задаються деякі глобальні критерії предметної області, якими він повинен керуватися при розкладанні карток на групи. В другому випадку, коли сформулювати глобальні критерії неможливо, експерту дається завдання розкласти картки на групи відповідно до інтуїтивного розуміння семантичної близькості запропонованих понять.

1.5. Методи встановлення семантичних відношень між поняттями предметної області

Останнім етапом побудови моделі предметної області є встановлення семантичних відношень між виділеними поняттями і метапоняттями. Встановити семантичні відношення – означає визначити специфіку взаємозв’язку, отриманого в результаті застосування тих чи інших методів. Для цього необхідно кожен зафіксований взаємозв’язок осмислити й віднести його до того чи іншого типу відношень.

Існує близько 200 базових відношень, наприклад, "частина – ціле", "рід – вид", "причина – наслідок", просторові, часові й інші відношення. Для кожної предметної області, крім загальних базових відношень, можуть існувати й унікальні відношення.

"*Прямий метод*" встановлення семантичних відношень базується на безпосередньому осмисленні кожного взаємозв’язку. У тому випадку, коли експерт не може дати інтерпретацію виділеного взаємозв'язку, йому пропонується така процедура. Формуються трійки "поняття № 1 – зв’язок – поняття № 2". Поруч із кожною трійкою записується коротке речення чи фраза, побудована так, щоб поняття № 1 і поняття № 2 входили в це речення. Як зв’язки використовуються тільки змістовні відношення (не використовують невизначені зв’язки «схожий на»).

Для „*непрямого методу*” необов’язково мати взаємозв’язки, досить лише наявності системи понять. Формулюється деякий критерій, для якого із системи понять обирається визначена

сукупність концептів. Ця сукупність пред'являється експерту з проханням дати вербальний опис сформульованого критерію. Концепти пред'являються експерту всі відразу (бажано на картках). У випадку ускладнень експерти звертаються до розподілу відібраних концептів на групи за допомогою більш дрібних критеріїв. Вихідна кількість концептів може бути довільною, але після розподілу на групи в кожній із таких груп повинно бути не більше десяти концептів. Після того, як складені описи по всіх групах, експерту пропонують об'єднати ці описи в один. Наступний крок у непрямому методі встановлення семантичних відношень – це аналіз тексту, складеного експертом. Концепти заміняють цифрами (це може бути вихідна нумерація), а зв'язки залишаються. Тим самим будується деякий граф, вершинами якого служать концепти, а дугами – зв'язки (наприклад, "через", "приводить до", "обумовлюючи", "сполучаючись", "визначає", "аж до" і т.д.) Цей метод дозволяє встановлювати не тільки базові відношення, але і відношення, специфічні для конкретної предметної області.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1. Ознайомитися з теоретичними відомостями.

2. Побудувати модель визначення предметної області, використовуючи всі методи побудови системи понять (1-6) та встановлення семантичних відношень між поняттями. Результуюча система понять має бути сумою понять, отриманих різними методами. Предметні області задано за варіантами (табл. 2.1).

Таблиця 2.1

Предметні області, які використовуються для побудови системи понять за варіантами *V*

V	Предметна область	*V*	Предметна область	*V*	Предметна область
1	Графічні редактори	11	Принтери	21	Антивіруси
2	Комп'ютерні мережі	12	Процесори	22	Інтернет
3	Комп'ютерні ігри	13	Сканери	23	Літаки
4	Математика	14	Жорсткі диски	24	Біологія
5	Медицина	15	Текстові редактори	25	Географія
6	Мови програмування	16	Телевізори	26	Хімія
7	Монітори	17	Електронні таблиці	27	Фізика
8	Операційні системи	18	Бази даних	28	Фрукти
9	Периферійні пристрої	19	Комп'ютерна графіка	29	Спорт
10	Комп'ютерні віруси	20	Цифрові фотоапарати	30	Телебачення

3. Звіт про виконання лабораторної роботи повинен містити розширену постановку задачі з описом предметної області та методів, які використовувались для побудови системи понять предметної області та встановлення семантичних відношень між ними. *Систему понять* оформити у вигляді таблиці, а *семантичні відношення* – у вигляді графа.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Опишіть будову і призначення експертних систем.
2. Як виконується визначення складу знань та їх розподіл в експертних системах?
3. Поясніть порядок побудови моделі предметної області.
4. Опишіть методи побудови системи понять предметної області.
5. Порівняйте методи встановлення семантичних відношень між поняттями предметної області.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер, 2000. – 384 с.
2. Глибовець М. М. Штучний інтелект / М. М. Глибовець, О. В. Олецький. – К. : КМ Академія, 2002. – 336 с.
3. Зайченко Ю. П. Основи проектування інтелектуальних систем: Навчальний посібник / Ю. П. Зайченко. – К. : Слово, 2004. – 352 с.
4. Локазюк В. М. Інтелектуальне діагностування мікропроцесорних пристроїв та систем: Навч. посібник для вузів / В. М. Локазюк, О. В. Поморова, А. О. Домінов. – Хмельницький – Київ : Такі справи, 2001. – 286 с.
5. Люгер Дж. Ф. Искусственный интеллект: Стратегии и методы решения сложных проблем / Дж. Ф. Люгер. – М. : Вильямс, 2003. – 864 с.
6. Системи штучного інтелекту : методичні вказівки до лабораторних робіт для студентів спеціальності "Комп’ютерні системи" / В. М. Локазюк, О. В. Поморова. – Хмельницький : ТУП, 2003. – 61 с.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БЗ – база знань
ГА – генетичний алгоритм
ЕС – експертна система
РО – розпізнавання образів
ШН – штучний нейрон
ШНМ – штучна нейронна мережа

ЗМІСТ

Навчальне видання

ЗАСОБИ ШТУЧНОГО ІНТЕЛЕКТУ В СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

Методичні вказівки до лабораторних робіт

Укладач Баловсяк Сергій Васильович

Відповідальний за випуск С. В. Мельничук

Літературний редактор О. В. Колодій

Підписано до друку 11.03.2014. Формат 60 х 84/16
Ум. друк. арк. 4,1.
Обл.-вид. арк. 4,4. Тираж 100
Видавничий дім „РОДОВІД"
58000, Чернівці, вул. Заводська, 26, а