**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД**
**«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»**

**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ**
**КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ І ПРОГРАМУВАННЯ**

**Єфіменко К.М., Добровольський Ю.М., Кучер Т.В.**

# КОНСПЕКТ ЛЕКЦІЙ

з нормативної навчальної дисципліни циклу природничо-наукової та загальноекономічної підготовки

# ІНФОРМАТИКА

для студентів всіх форм навчання

Галузь знань: 0305 Економіка та підприємництво

Напрям підготовки: 6.030508 Фінанси і кредит

Спеціалізація: *Фінанси підприємства* (ФПР), *Банківська справа* (ФБС), *Фінансова аналітика* (ФА), *Оподаткування* (ФП)

Розглянуто
на засіданні кафедри ОМіП
протокол № 1 від "30" серпня 2013 р.

Затверджено
навчально-видавничою радою ДонНТУ
протокол № _ від "__" _____ 201_ р.

**2013**

**УДК 004.432.2**

Конспект лекцій з нормативної навчальної дисципліни циклу природничо-наукової та загальноекономічної підготовки «Інформатика» для студентів всіх форм навчання галузі знань: 0305 «Економіка та підприємництво», напряму підготовки: 6.030508 «Фінанси і кредит», спеціалізації: «Фінанси підприємства», «Банківська справа», «Фінансова аналітика», «Оподаткування»/ Укл.: К.М. Єфіменко, Ю.М. Добровольський, Т.В. Кучер. – Донецьк: ДВНЗ ДонНТУ. – 2013. – 203 с.

Конспект лекцій містить теоретичний матеріал згідно вимогам освітньо-професійної програми підготовки спеціалістів галузі знань 0305 «Економіка та підприємництво».

***Автори:*** К.М. Єфіменко, к.т.н., доцент

Ю.М. Добровольський, ст. викладач

Т.В. Кучер, асистент

***Відп. за випуск***: В.М. Павлиш, д.т.н., професор

# ЗМІСТ

# Ціль і задачі курсу

Термін ***інформатика*** виник в 60-х роках XX ст. у Франції для назви галузі науки, що займається автоматизованою обробкою інформації за допомогою електронних обчислювальних машин. Французький термін утворений шляхом злиття слів «інформація» і «автоматика» і означає «інформаційна автоматика або автоматизована переробка інформації». В англомовних країнах цьому терміну відповідає синонім ***computer science*** (наука про комп'ютерну техніку).

**Інформатика** – це область людської діяльності, пов'язана із процесами перетворення інформації за допомогою комп'ютерів і їхньою взаємодією із середовищем застосування. Інформатика займається вивченням процесів перетворення й створення нової інформації більш широко, практично не вирішуючи завдання керування різними об'єктами, як кібернетика. Інформатика з'явилася завдяки розвитку комп'ютерної техніки, базується на ній і зовсім немислима без неї.

Головна функція інформатики полягає в розробці методів і засобів перетворення інформації і їхньому використанню в організації технологічного процесу переробки інформації.

**Ціль курсу** «Інформатика» – формування у студентів навичок роботи на комп'ютері та алгоритмічного мислення, уміння здійснювати постановку задачі для розробки програмного забезпечення і реалізації алгоритмів у вигляді комп'ютерних програм.

**Задачі курсу** – вивчення принципів роботи з операційною системою комп'ютера і організації обчислювальних процесів, принципів алгоритмізації, основних типів алгоритмів, способів їхнього представлення, освоєння етапів розробки програм.

Для програмної реалізації алгоритмів пропонується використати мову Visual Basic for Application (VBA), убудовану у додатки пакета Microsoft Office і яка поширює їх стандартні можливості.

У результаті вивчення курсу студент повинен:

- вміти користуватися Microsoft Windows і пакетом Microsoft Office;
- знати типи алгоритмів і етапи розробки програм;
- вміти застосовувати Internet при розв’язанні фахових завдань.

Курс складається із сімнадцяти лекцій (одна лекція приділяється для самостійного вивчення) і шістнадцяти лабораторних робіт. Кожна лабораторна робота виконується в наступному порядку:

1. Одержання індивідуального завдання відповідно до варіанта студента.
2. Розв'язання поставленої задачі.
3. Оформлення звіту.
4. Подання звіту викладачу.

# Лекція №1. СИСТЕМИ ЧИСЛЕННЯ

## 1.1. Позиційні системи числення

Із древніх часів людство користується числами й операціями над ними. При цьому найбільше розповсюдження в повсякденному житті отримала десяткова система числення (с/ч), у якій використовуються десять арабських цифр від 0 до 9. Розташовуючи цифри в різних позиціях, ми отримуємо різні числа. Така с/ч називається позиційною, у ній величина числа визначається положенням і значенням кожної його цифри. Іншим прикладом позиційної с/ч може служити римська система числення. Поряд з позиційними існують і непозиційні системи числення. Наприклад, такою с/ч користувався Робінзон Крузо, що за допомогою карбів відзначав кількість проведених на незаселеному острові днів.

Будь-яке число N у позиційній системі числення можна представити в наступному вигляді:

$$N = (b_m\, b_{m-1}\ldots b_2 b_1 b_0,\, b_{-1} b_{-2}\ldots b_{-k})_A, \qquad (1.1)$$

де b – цифра з алфавіту с/ч;

, – роздільник цілої й дробової частини числа;

A – основа системи числення (для десяткової с/ч A = 10);

m ... k – вагарні коефіцієнти.

Наприклад, десяткове число $205{,}74_{10}$ відповідно до (1.1) має наступні вагові коефіцієнти: $2^2\ 0^1\ 5^0$ , $7^{-1}\ 4^{-2}$.

Використовуючи ті ж позначення, число N можна представити у вигляді суми елементів ряду:

$$N = b_m \cdot A^m + b_{m-1} \cdot A^{m-1} + \ldots + b_1 \cdot A^1 + b_0 \cdot A^0 + b_{-1} \cdot A^{-1} + \ldots + b_{-k} \cdot A^{-k} \ldots \qquad (1.2)$$

Наприклад: $205{,}74_{10} \rightarrow 2^2 0^1 5^0, 7^{-1} 4^{-2} \rightarrow 2 \cdot 10^2 + 0 \cdot 10^1 + 5 \cdot 10^0 + 7 \cdot 10^{-1} + 4 \cdot 10^{-2}$

Наведений вираз (1.2), є універсальним для будь-якої позиційної системи числення.

Незважаючи на те, що десяткова с/ч є для нас найбільш звичною й зручною у використанні, реалізація на її основі обчислювальній техніці не раціонально. Тому основною системою числення для внутрішнього зберігання й обробки даних у комп'ютері є **двійкова** й похідні від неї восьмерична й шістнадцятерична с/ч.

### 1.1.1. Двійкова система числення

У двійковій системі числення (2 с/ч) використовується дві цифри 0 і 1, основа с/ч A=2. Наприклад, двійкове число $101101_2$ відповідає десятковому числу $45_{10}$.

Використання у двійковій с/ч мінімальної кількості цифр, для запису чисел, дозволяє найбільше економічно реалізовувати апаратну частину ЕОМ. Ко-

жна цифра двійкового числа називається ***біт***. Біт називається також двійковим розрядом. Група з 8 біт складає ***байт***, що може зберігати різні типи даних, такі як літери алфавіту, десяткові цифри або інші знаки. Таким чином, **1 біт = $= 2^3$ байт**.

Байт є основною одиницею виміру інформації. Крім цього для виміру обсягу інформації часто використовуються наступні похідні від байта:

1 Кбайт (кілобайт) = 1024 байт = $2^{10}$ байт,

1 Мбайт (мегабайт) = 1024 Кбайт = $2^{20}$ байт,

1 Гбайт (гігабайт) = 1024 Мбайт = $2^{30}$ байт,

1 Тбайт (терабайт) = 1024 Гбайт = $2^{40}$ байт.

Для конвертування чисел із двійкової с/ч у десяткову використовуються вирази (1.1, 1.2).

**Приклад**. $11101{,}01_2 \rightarrow (?)_{10}$, послідовність вагових коефіцієнтів має вигляд $1^4\,1^3\,1^2\,0^1\,1^0,\ 0^{-1}\,1^{-2} \rightarrow 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 0\cdot2^{-1} + 1\cdot2^{-2} = 29{,}25_{10}$

Однак істотним недоліком двійкової с/ч є громіздкий запис чисел. Для спрощення запису двійкових чисел можуть бути використані восьмерична або шістнадцятерична системи числення.

### 1.1.2. Восьмерична система числення

У восьмеричній системі числення, що є похідною від двійкової, використовується вісім цифр від 0 до 7, і її основа A=8. Основу восьмеричної с/ч, тобто число 8, можна представити у вигляді $2^3$. Тому одній восьмеричній цифрі відповідає три двійкових розряди – **тріада**.

Відповідність між восьмеричним числом і його двійковим і десятковим представленнями наведено у таблиці 1.1.

Таблиця 1.1

Зв'язок між восьмеричною, двійковою й десятковою с/ч

| Восьмерична с/ч | Двійкова с/ч | Десяткова с/ч |
|---|---|---|
| 0 | 000 | 0 |
| 1 | 001 | 1 |
| 2 | 010 | 2 |
| 3 | 011 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |
| 10 | 001000 | 8 |
| 11 | 001001 | 9 |
| 12 | 001010 | 10 |

Для конвертування двійкового числа у восьмеричне необхідно розбити двійкове число на тріади в такий спосіб: ціла частина розбивається на тріади, починаючи з молодших розрядів (із правого краю цілої частини числа), а дробова частина - з лівого краю. Розряди, яких не вистачає для формування тріад з лівого або правого країв, заповнюються 0. Отримані тріади за допомогою таблиці 1.1, заміняються восьмеричними цифрами.

**Приклад**. $1011010110{,}01_2 \rightarrow (?)_8$

$1011010110{,}01_2 \rightarrow \underline{\mathbf{00}}\mathbf{1}\ \underline{011}\ \underline{010}\ \underline{110}\ ,\ \underline{01}\mathbf{0} \rightarrow 1326{,}2_8$

Для виконання зворотного переведення (з 8 с/ч в 2 с/ч) кожну восьмеричну цифру заміняють відповідною двійковою тріадою. Незначні 0 у цілій і дробовій частинах отриманого числа можна відкинути.

**Приклад.** $357{,}24_8 \rightarrow (?)_2$

$357{,}24_8 \rightarrow \underline{011}\ \underline{101}\ \underline{111}\ ,\ \underline{010}\ \underline{100} \rightarrow 11101111{,}0101_2$

Для конвертування числа з восьмеричної с/ч у десяткову також використовуються вирази (1.1, 1.2).

**Приклад**. $357{,}24_8 \rightarrow (?)_{10}$

послідовність вагових коефіцієнтів має вигляд $3^2\ 5^1\ 7^0{,}2^{-1}\ 4^{-2} \rightarrow$

$3\cdot 8^2 + 5\cdot 8^1 + 7\cdot 8^0 + 2\cdot 8^{-1} + 4\cdot 8^{-2} = 239{,}3125_{10}$

### 1.1.3. Шістнадцятерична система числення

У шістнадцятеричної системі числення використовується десять цифр від 0 до 9 і шість латинських літер A, B, C, D, E, F. Основу с/ч A=16 можна представити у вигляді $2^4$. Тому однієї шістнадцятеричній цифрі відповідає чотири двійкових розряди – **тетрада**.

Відповідність між шістнадцятеричним числом і його двійковим і десятковим представленнями наведено у таблиці 1.2.

Таблиця 1.2

Зв'язок між шістнадцятеричною, двійковою й десятковою с/ч.

| Шістнадцятерична с/ч | Двійкова с/ч | Десяткова с/ч | Шістнадцятерична с/ч | Двійкова с/ч | Десяткова с/ч |
|---|---|---|---|---|---|
| 0 | 0000 | 0 | 8 | 1000 | 8 |
| 1 | 0001 | 1 | 9 | 1001 | 9 |
| 2 | 0010 | 2 | A | 1010 | 10 |
| 3 | 0011 | 3 | B | 1011 | 11 |
| 4 | 0100 | 4 | C | 1100 | 12 |
| 5 | 0101 | 5 | D | 1101 | 13 |
| 6 | 0110 | 6 | E | 1110 | 14 |
| 7 | 0111 | 7 | F | 1111 | 15 |
| | | | 10 | 00010000 | 16 |

Конвертування із двійкової с/ч у шістнадцятеричну с/ч і зворотно виконується способом, описаним для восьмеричної с/ч. Отримані при цьому тетради за допомогою таблиці 1.2, заміняються шістнадцятеричними цифрами й навпаки.

**Приклад:** $1011010110{,}01_2 \rightarrow (?)_{16}$

$1011010110{,}01_2 \rightarrow \underline{\mathbf{00}10}\ \underline{1101}\ \underline{0110}\ ,\ \underline{0\mathbf{100}} \rightarrow 2D6{,}2_{16}$

$4A9{,}B2_{16} \rightarrow (?)_2$

$4A9{,}B2_{16} \rightarrow \underline{0100}\ \underline{1010}\ \underline{1001}\ ,\ \underline{1011}\ \underline{0010} \rightarrow 10010101001{,}1011001_2$

При конвертуванні із шістнадцятеричної с/ч у десяткову також використовуються вирази (1.1, 1.2).

**Приклад:** $2D6{,}2_{16} \rightarrow (?)_{10}$

послідовність вагових коефіцієнтів має вигляд $2^2\ D^1\ 6^0{,}2^{-1} \rightarrow$

$2\cdot16^2 + 13\cdot16^1 + 6\cdot16^0 + 2\cdot16^{-1} = 726{,}125_{10}$

Переведення із шістнадцятеричної с/ч у восьмеричну й навпаки виконується у два етапи. На початку виконується конвертування у двійкову с/ч, а потім із двійкової в необхідну с/ч, описаним вище способом.

**Приклад:** $4A9{,}B_{16} \rightarrow (?)_8$

$4A9{,}B_{16} \rightarrow \underline{0100}\ \underline{1010}\ \underline{1001{,}1011_2} \rightarrow \underline{010}\ \underline{010}\ \underline{101}\ \underline{001{,}101}\ \underline{100_2} \rightarrow 2251{,}54_8$

$357{,}24_8 \rightarrow (?)_{16}$

$357{,}24_8 \rightarrow \underline{011}\ \underline{101}\ \underline{111{,}010}\ \underline{100_2} \rightarrow \underline{0000}\ \underline{1110}\ \underline{1111{,}0101}\ \underline{0000_2} \rightarrow EF{,}5_{16}$

## 1.2. Переведення цілої частини десяткового числа у різні системи числення

Переведення цілого числа представленого в десяткової с/ч у двійкову, восьмеричну або шістнадцятеричну системи числення виконується шляхом цілочисельного ділення вхідного числа на основу нової с/ч. При цьому необхідно виконати наступну послідовність дій:

1. Розділити націло вхідне число на основу нової с/ч. Остача від ділення дає молодшу цифру числа в нової с/ч.

2. Якщо частка від ділення не дорівнює 0, то перейти до пункту 3, інакше перейти до пункту 4.

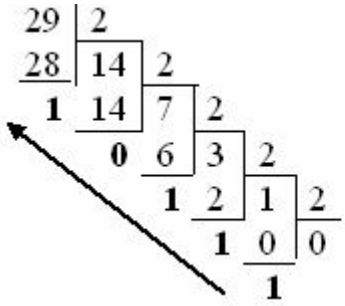
3. Розділити отриману частку націло на основу нової с/ч. Остача від ділення дає наступну цифру числа в нової с/ч. Перейти до пункту 2.

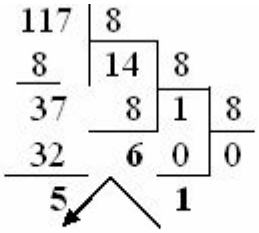
4. Отримані в результаті ділень остачі, записати в порядку зворотному їхньому обчисленню. Це й буде вхідне число в нової с/ч.

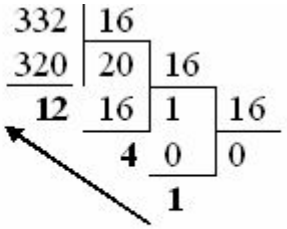
Особливу увагу варто приділити переведенню в шістнадцятеричну с/ч, тому що залишки від цілочисельного ділення в цьому випадку, можуть перевищувати число 9, тому при записі кінцевого результату, важливо не забувати заміняти їх відповідною шістнадцятеричною цифрою (позначуваною латинськи-

ми літерами від A до F).

**Приклад:** $29_{10} \rightarrow (?)_2$ $117_{10} \rightarrow (?)_8$ $332_{10} \rightarrow (?)_{16}$




$29_{10} \rightarrow 11101_2$ $117_{10} \rightarrow 165_8$ $332_{10} \rightarrow 14C_{16}$

## 1.3. Переведення дробової частини десяткового числа в різні системи числення із заданою точністю

Переведення дробової частини числа представленого в десятковій с/ч у двійкову, восьмеричну або шістнадцятеричну системи числення виконується шляхом множення дробової частини вхідного числа на основу нової с/ч. Тому що процес послідовного множення може тривати нескінченно, то переведення виконується або до одержання необхідної кількості розрядів у дробовій частині числа в нової с/ч або до досягнення заданої точності. При цьому необхідно виконати наступну послідовність дій:

1. Помножити дробову частину вхідного числа на основу нової с/ч. Ціла частина отриманого добутку дає першу цифру дробової частини числа в нової с/ч.

2. Дробову частину отриманого добутку помножити на основу нової с/ч. Ціла частина отриманого добутку дає наступну цифру дробової частини числа в нової с/ч.

3. Якщо досягнуто задану точність або отримано необхідну кількість цифр у дробовій частині числа в нової с/ч те перейти до п. 4, інакше повторити п. 2.

4. Отримані в результаті множення цілі частини добутків, записати в порядку їхнього обчислення. Це й буде дробова частина вхідного числа в нової с/ч.

**Приклад**. Виконати переведення дробової частини десяткового числа у двійкову, восьмеричну й шістнадцятеричну с/ч, як критерій використовуючи, необхідність одержати 3 цифри в дробовій частині числа в нової с/ч.

$0{,}74_{10} \rightarrow (?)_2$ $0{,}23_{10} \rightarrow (?)_8$ $0{,}12_{10} \rightarrow (?)_{16}$

| | | |
|---|---|---|
| 0,74 | 0,23 | 0,12 |
| 2 | 8 | 16 |
| 1, 48 | 1, 84 | 1, 92 |
| 2 | 8 | 16 |
| 0, 96 | 6, 72 | 14, 72 |
| 0, 96 | 6, 72 | 14, 72 |
| 2 | 8 | 16 |
| 1, 92 | 5, 76 | 11, 52 |

Отримані наступні результати:

$0{,}74_{10} \to 0{,}101_2$ $0{,}23_{10} \to 0{,}165_8$ $0{,}12_{10} \to 0{,}1EB_{16}$

Переведемо отримані значення назад в десяткову систему числення й визначимо $\Delta_A$ погрішність даного способу переведення.

$0{,}101_2 \to 0^0, 1^{-1}\,0^{-2}\,1^{-3} = 0\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = 0{,}625_{10}$

Погрішність для двійкової с/ч складе $\Delta_2 = 0{,}74 - 0{,}625 = 0{,}115$.

$0{,}165_8 \to 0^0, 1^{-1}\,6^{-2}\,5^{-3} = 0\cdot8^0 + 1\cdot8^{-1} + 6\cdot8^{-2} + 5\cdot8^{-3} \approx 0{,}229_{10}$

Погрішність для восьмеричної с/ч складе $\Delta_8 = 0{,}23 - 0{,}229 = 0{,}001$.

$0{,}1EB_{16} \to 0^0, 1^{-1}\,E^{-2}\,B^{-3} = 0\cdot16^0 + 1\cdot16^{-1} + 14\cdot16^{-2} + 11\cdot16^{-3} \approx 0{,}1199_{10}$

Погрішність для шістнадцятеричної с/ч складе $\Delta_{16}$=0,12–0,1199=0,0001.

Проаналізувавши отримані результати, можна зробити висновок: чим менше інформаційна ємність системи числення, тим більше погрішність переведення. Отже, використаний критерій переведення не придатний для виконання розрахунків.

Як відомо, **точність**, з якої задане число, визначається кількістю цифр у його дробовій частині.

Для десяткової с/ч точність представлення числа визначається в такий спосіб:

$(0, b_{-1}\, b_{-2} \ldots b_{-k})_{10} \to b_{-1}\cdot10^{-1} + b_{-2}\cdot10^{-2} + \ldots + b_{-k}\cdot10^{-k}$

Звідси:

точність, що дає цифра з ваговим коефіцієнтом -1 дорівнює $10^{-1} = 0.1$;
точність, що дає цифра з ваговим коефіцієнтом -2 дорівнює $10^{-2} = 0.01$;
точність, що дає цифра з ваговим коефіцієнтом -k дорівнює $10^{-k}$.

У загальному випадку, точність, з якої задається число, у будь-якій позиційній системі числення визначається з виразу

$$t_A = \frac{1}{A^R}, \qquad (1.3)$$

де A – основа системи числення;

R – кількість цифр у дробовій частині числа.

Точність переведення задається в десятковій системі числення ($t_{10}$). Для виконання переведення із заданою точністю необхідно одержати таку кількість

цифр у дробовій частині числа в новій системі числення А, щоб $t_{10} > t$.

**Приклад.** Виконати переведення $0{,}74_{10} \rightarrow (?)_2$ з точністю $t = 0{,}01$.

0, 74
× 2
1, 48 точність, що дає ця цифра $t_2 = 2^{-1} = 0{,}5 > t = 0{,}01$;
× 2
0, 96 точність, що дає ця цифра $t_2 = 2^{-2} = 0{,}25 > t = 0{,}01$;
× 2
1, 92 точність, що дає ця цифра $t_2 = 2^{-3} = 0{,}125 > t = 0{,}01$;
× 2
1, 84 точність, що дає ця цифра $t_2 = 2^{-4} = 0{,}0625 > t = 0{,}01$;
× 2
1, 68 точність, що дає ця цифра $t_2 = 2^{-5} = 0{,}03125 > t = 0{,}01$;
× 2
1, 36 точність, що дає ця цифра $t_2 = 2^{-6} \approx 0{,}01563 > t = 0{,}01$;
× 2
0, 72 точність, що дає ця цифра $\mathbf{t_2 = 2^{-7} \approx 0{,}00781 < t = 0{,}01}$;

$0{,}74_{10} \rightarrow 0{,}1^{-1}\, 0^{-2}\, 1^{-3}\, 1^{-4}\, 1^{-5}\, 1^{-6}\, 0^{-7} = 0{,}1011110_2$

Отримана точність представлення числа у двійкової с/ч $t_2$ менше заданої точності $t_{10}$, отже, процес переведення можна завершити. Визначимо погрішність переведення, попередньо виконавши зворотне переведення:

$0{,}1011110_2 \rightarrow 0^0, 1^{-1}\, 0^{-2}\, 1^{-3}\, 1^{-4}\, 1^{-5}\, 1^{-6}\, 0^{-7} =$

$= 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 1 \cdot 2^{-6} + 0 \cdot 2^{-7} \approx 0{,}734_{10}$

Погрішність склала $\Delta_2 = 0{,}74 - 0{,}734 = 0{,}006$ і не перевищує $t_{10}$.

**Приклад.** Виконати переведення $0{,}23_{10} \rightarrow (?)_8$ з точністю $t = 0{,}001$.

0,23 $0{,}23_{10} \rightarrow 0{,}1^{-1}\, 6^{-2}\, 5^{-3}\, 6^{-4} = 0{,}1656_8$
× 8
1, 84 точність, що дає ця цифра $t_8 = 8^{-1} = 0{,}125 > t = 0{,}001$;
× 8
6, 72 точність, що дає ця цифра $t_8 = 8^{-2} = 0{,}015625 > t = 0{,}001$;
× 8
5, 76 точність, що дає ця цифра $t_8 = 8^{-3} \approx 0{,}00195 > t = 0{,}001$;
× 8
6, 08 точність, що дає ця цифра $\mathbf{t_8 = 8^{-4} \approx 0{,}00024 < t = 0{,}001}$;

Отримана точність представлення числа у восьмеричної с/ч $t_8$ менше заданої точності $t_{10}$, отже, процес переведення можна завершити. Визначимо по-

грішність переведення, попередньо виконавши зворотне переведення:

$0{,}1656_8 \rightarrow 0^0, 1^{-1}\ 6^{-2}\ 5^{-3}\ 6^{-4} = 0\cdot8^0 + 1\cdot8^{-1} + 6\cdot8^{-2} + 5\cdot8^{-3} + 6\cdot8^{-4} \approx 0{,}22998_{10}$

Погрішність склала $\Delta_8 = 0{,}23 - 0{,}22998 = 0{,}00002$ і не перевищує $t_{10}$.

Переведення десяткового числа, що містить цілу й дробову частини, у двійкову, восьмеричну або шістнадцятеричну системи числення, відбувається у два етапи. Спочатку переводиться ціла частина, а потім дробова частина числа.

### 1.4. Завдання для самостійної роботи

Послідовно виконати переведення чисел у зазначені системи числення (переведення з десяткової с/ч виконати із заданою точністю):

1. $1011001{,}01011_2 \rightarrow (?)_{10} \rightarrow (?)_8$ (точність переведення $t = 0{,}002$) $\rightarrow (?)_2$.
2. $546{,}26_8 \rightarrow (?)_{10} \rightarrow (?)_2$ (точність переведення $t = 0{,}01$) $\rightarrow (?)_8$.
3. $1AE{,}38_{16} \rightarrow (?)_{10} \rightarrow (?)_8$ (точність переведення $t = 0{,}0005$) $\rightarrow (?)_2 \rightarrow (?)_{16}$.
4. $1101001101010.1001_2 \rightarrow (?)_8 \rightarrow (?)_{16} \rightarrow (?)_2$.
5. $1011101110.01011_2 \rightarrow (?)_{16} \rightarrow (?)_8 \rightarrow (?)_2$.
6. $63417{,}464_8 \rightarrow (?)_{16} \rightarrow (?)_8$.
7. $F09C{,}08D6_{16} \rightarrow (?)_8 \rightarrow (?)_{16}$.
8. $94{,}37_{10} \rightarrow (?)_2$ (точність переведення $t = 0{,}04$) $\rightarrow (?)_{16} \rightarrow (?)_{10}$.
9. $238{,}512_{10} \rightarrow (?)_8$ (точність переведення $t = 0{,}00025$) $\rightarrow (?)_2 \rightarrow (?)_{10}$.
10. $493{,}951_{10} \rightarrow (?)_{16}$ (точність переведення $t = 0{,}00001$) $\rightarrow (?)_8 \rightarrow (?)_{10}$.

## Лекція №2. ОПЕРАЦІЙНА СИСТЕМА MICROSOFT WINDOWS

### 2.1. Загальні теоретичні положення

**Персональний комп'ютер** (ПК) являє собою сукупність апаратної частини й програмного забезпечення, що управляє роботою апаратної частини. Операційна система складає основу будь-якого встановленого на ПК програмного забезпечення.

***Операційна система*** (ОС) – це сукупність програмних засобів, що забезпечують виконання наступних основних функцій:

– керування апаратною частиною ПК;

– запуск і виконання прикладного програмного забезпечення;

– підтримка користувальницького ***інтерфейсу*** – набору засобів і способів спілкування користувача із ПК.

У цей час найбільше поширення отримали операційні системи сімейств **Windows, Linux, MacOs** і **Android**. ОС сімейства Microsoft Windows, MacOs розповсюджуються на комерційній основі, у те час як ОС сімейства Linux є вільно розповсюджуваним програмним продуктом. При цьому кожна ОС у порі-

внянні з іншими має як свої переваги, так і недоліки.

Лідируючі позиції серед ОС сімейства Microsoft Windows займають Windows XP, Windows 7 і Windows 8. Головні принципи роботи в будь-якій версії ОС Microsoft Windows залишаються незмінними, починаючи з версії Windows 95, що значно полегшує процес навчання.

## 2.2. Файлова система ОС Windows

Існує два основних способи організації файлової системи ПК, що працює під управлінням ОС Microsoft Windows: FAT і NTFS. ***Файлова система*** ОС Windows включає 4 основні об'єкти: диски, файли, папки і ярлики.

### 2.2.1. Диски

Всі дані в ПК зберігаються на дисках. Жорсткий магнітний диск ПК («вінчестер») звичайно ділиться на кілька окремих частин – ***локальні диски***. Кожен диск має своє *ІМ'Я*, що складається з латинської літери й «:». Імена локальних дисків починаються з літери «С». Наприклад, ПК може мати наступні диски:

**A:** – дисковод для роботи з дискетами 3.5';

**C:** – 1-й локальний диск, на якому звичайно записана ОС («системний» диск);

**D:** – 2-й локальний диск, на якому звичайно зберігаються дані користувача («робочий» диск);

**E:** – дисковод для роботи з DVD і CD дисками.

Будь-який зовнішній пристрій, що підключає до ПК який має убудовану пам'ять (телефон або фотоапарат з картою пам'яті, «флешка» і т.д.), ідентифікується як зовнішній диск, і позначається першою вільною латинською літерою, що йде після імен стаціонарних дисків.

### 2.2.2. Файли

Всі дані на дисках зберігаються у вигляді файлів. ***Файл*** – це будь-які дані, що записані на диск під визначеним іменем. Повне ***ім'я файлу*** складається із двох частин: імені й розширення, які при записі розділяються крапкою. Ім'я файлу може містити до 255 різних символів, за винятком ***, ?, :, <, >, \, /,** |. Розширення є необов'язковою частиною повного імені, звичайно складається із трьох або чотирьох латинських літер, і визначає тип файлу, тобто вказує якого типу дані зберігаються в цьому файлі.

Наприклад:

**Звіт.doc**

де «Звіт» – ім'я файлу, «doc» – розширення.

Умовно всі файли можна розділити на 3 типи:

1. *Файли-документи*, які містять дані, введені користувачем (текст, графіка, аудіо, відео). Такі файли мають наступні стандартні розширення:

.txt, .doc – текстові файли;

.xls – файли з електронними таблицями;

.bmp, .jpg – графічні файли;

.mp3, .wma – аудіо-файли;

.avi, .wmv, vob – відео-файли.

2. *Файли-програми*, які містять готові до виконання програмні коди й мають розширення .exe, .com. Програмне забезпечення (ПО) ділиться на 2 види: системне ПО (Hard wave) – це програми, що управляють роботою апаратної частини ПК; прикладне ПО (Soft wave) – це програми, які призначені для роботи користувачів (редактори, програвачі, конвектори, ігри та ін.).

3. *Допоміжні файли*, які забезпечують роботу програм. Сукупність основної програми й допоміжних файлів утворить ***додаток***.

*Шаблони імен файлів.* Якщо необхідно виконати яку-небудь дію не з одним файлом, а із їхньою групою (наприклад, знайти), то замість конкретного ім'я файлу можна вказувати шаблон (маску). Для завдання шаблона використовуються наступні символи:

* – позначає, що замість неї в імені файлу може стояти будь-яка кількість будь-яких символів.

? – позначає один будь-який символ в імені файлу.

Наприклад:

*.doc – всі файли (з будь-яким іменем) з розширенням doc.

a*.txt – всі файли з розширенням txt, ім'я яких починається на літеру «а».

???.doc – всі файли з розширенням doc, ім'я яких складається максимум із трьох символів.

*.* – всі файли.

### 2.2.3. Папки

***Папка*** – це пойменована область диска, використовувана для зберігання файлів. Ім'я папки задається по тимі ж правилам, що й ім'я файлу, тільки без розширення. Папки впорядковують розташування файлів на дисках і прискорюють процес їхнього пошуку. Папки можуть бути вкладеними одна в іншу. При цьому папка стосовно вкладеного в неї папками є «батьківської». Папки діляться на системні (*Мій комп'ютер*, *Робочий стіл* та ін.) і звичайні. *Системні папки* – це папки, які являється складовою частиною ОС і не можуть бути вилучені користувачем.

Структуру даних на диску зручно представляти у виді ***дерева папок***, гілки якого утворюються вкладеними папками (рис. 2.1).

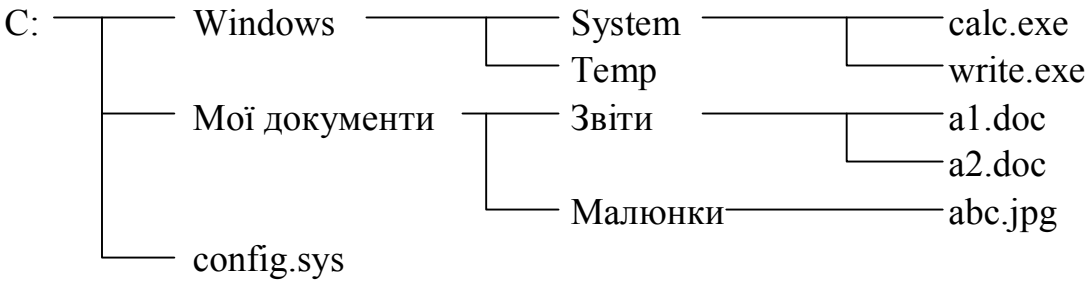



Рисунок 2.1. Дерево папок

Для зазначення місця розташування об'єкта на диску необхідно задати ***повний шлях*** до нього (адреса). Шлях починається з імені диска й містить послідовність вкладених папок, які розділені \. Шлях читається праворуч ліворуч.

Наприклад, шлях до файлу write.exe наступний

C:\Windows\System\write.exe

Файл write.exe розташований у папці System, яка знаходитися в папці Windows на диску C:.

Шлях до файлу a2.doc наступний

C:\Мої документи\Звіти\a2.doc

Файл a2.doc розташований у папці Звіти, яка знаходитися в папці Мої документи на диску C:.

### 2.2.4. Ярлики

***Ярлик*** – це спеціальний файл, що містить повний шлях до об'єкта для якого він створений. Ярлики мають розширення .lnk і використовуються для швидкого відкриття дисків, файлів і папок.

Кожний об'єкт файлової системи зображений на екрані у вигляді спеціального значка (***піктограми***) (мал. 2.2).

Локальний диск | Диск 3.5 | CD, DVD диск | Папка | Файл | Ярлик

Рисунок 2.2. Піктограми об'єктів

## 2.3. Елементи управління. Структура вікна папки

Після завантаження ОС екран має такий вигляд (рис. 2.3):

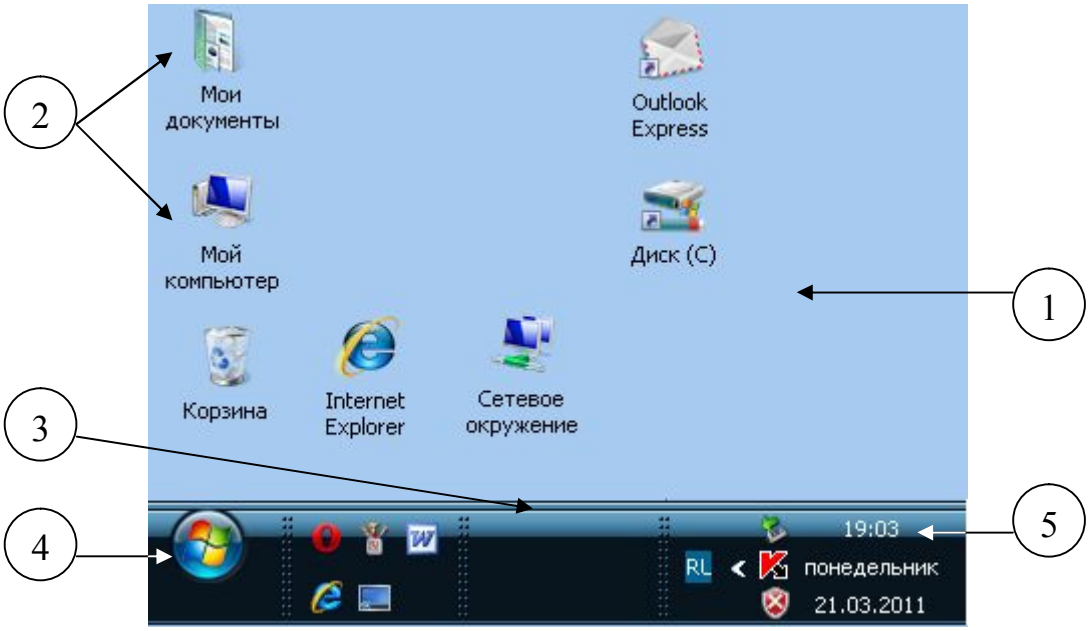



Рисунок 2.3. Вікно ОС Windows

Основну частину екрана займає «Робочий стіл» (1), на якому розташовані значки системних папок (2) і інших об'єктів. Нижче робочого стола розташована «Панель задач» (3), на якій знаходиться кнопка «Пуск» (4), що відкриває головне меню Windows, а так само індикатори часів і клавіатури (5).

### 2.3.1. Елементи управління

Основними елементами управління, що утворять інтерфейс користувача є:

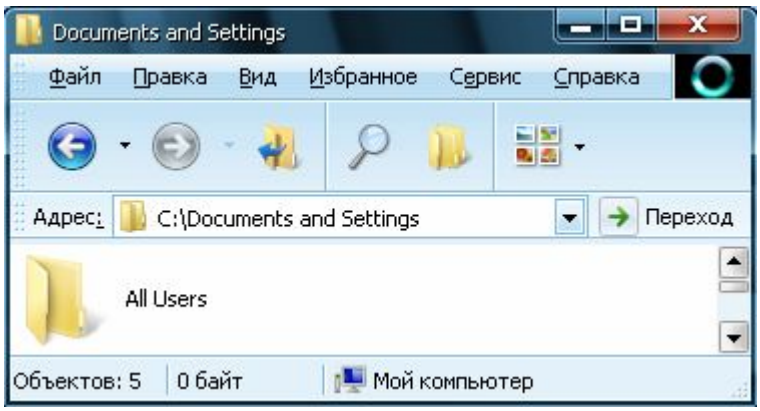
1. ***Вікно*** – призначено для відображення різних видів інформації. Вікна можуть бути трьох типів:

- вікно папки (рис.2.4 а);
- діалогове вікно (рис.2.4 б);
- інформаційне вікно (рис.2.4 в).

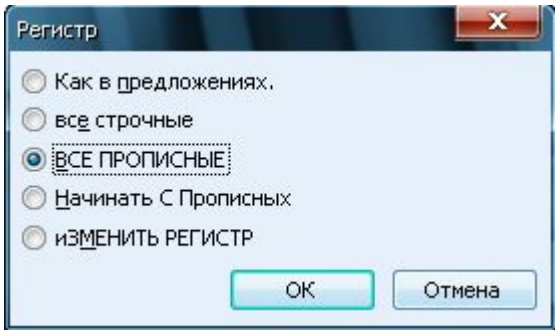
2. ***Меню*** призначено для виконання команд і запуску програм (рис. 2.5). Меню складається з окремих пунктів. Пункт меню може виконувати зазначену команду або відкривати підменю (містить знак ▸). Для вибору пункту меню або відкриття підміню досить підвести на нього курсор миші. Для виконання пункту меню необхідно виконати одинарного щиглика лівої кнопки миші (1 лкм) на ньому.

3. ***Кнопка*** (рис. 2.6) призначена для виконання визначених команд. Для натискання кнопки виконати 1 лкм на ній.

а)




б)




в)

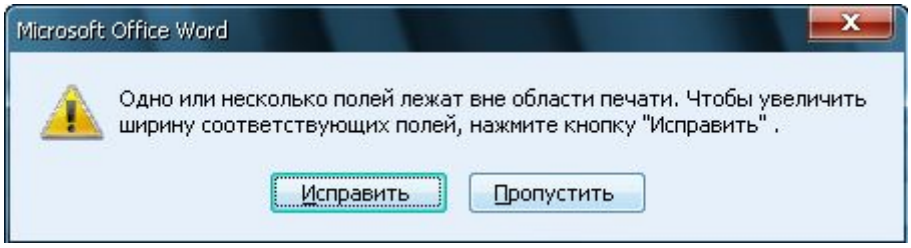



Рисунок 2.4. Види вікон ОС Windows

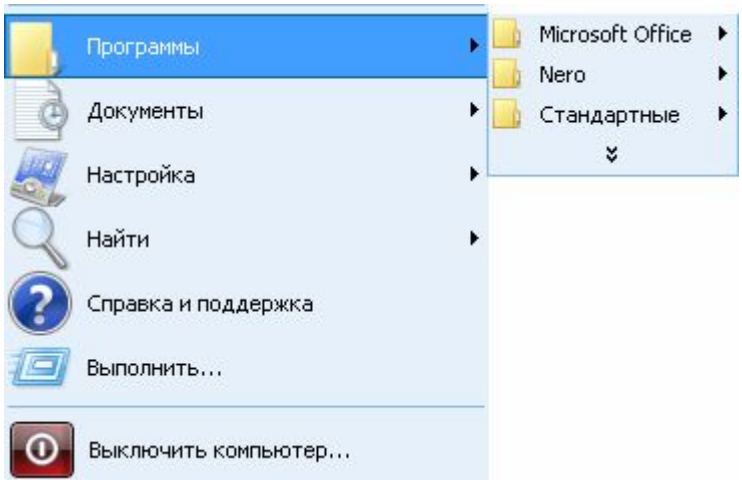



Рисунок 2.5. Зовнішній вигляд класичного меню

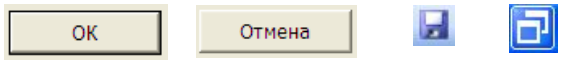


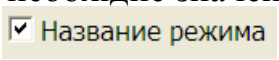

Рисунок 2.6. Кнопки

4. ***Текстове поле***. 

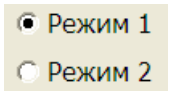
Призначено для вводу даних у діалогових вікнах.

5. ***Текстове поле із списком***. 

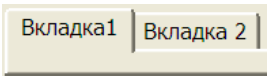
Дозволяє ввести необхідне значення, або вибрати його зі списку.

6. ***Прапорець***. 


Призначений для вмикання/вимикання зазначеного режиму.

7. ***Перемикачі***. 


Звичайно розташовуються в групі й призначені для вибору (вмикання) тільки одного з перерахованих режимів.

8. ***Вкладки***. 


Призначені для вибору різних сторінок діалогового вікна.

### 2.3.2. Структура головного меню Windows (класична)

Кнопка «Пуск» відкриває головне меню ОС Windows. Класичне меню «Пуск» включає наступні розділи:

1. ***Программы*** – призначений для запуску на виконання встановлених на ПК програм.

2. ***Документы*** – містить список останніх документів, що відкривались, і призначений для їх швидкого повторного відкриття.

3. ***Настройка*** – призначені для настроювання параметрів ОС Windows.

4. ***Найти*** – призначений для пошуку файлів і папок на дисках.

5. ***Выключить компьютер*** – призначений для регламентованого завершення роботи із ПК.

### 2.3.3. Структура вікна папки

Для роботи з дисками, папками або файлами необхідно відкрити системну папку «Мій комп'ютер», що містить список наявних на ПК дисків. Папка відкривається у вигляді вікна, у якому відображається вміст папки. Вікно папки має наступну структуру (рис. 2.7):

1 – рядок заголовка вікна, у якій відображається ім'я відкритого об'єкта.

2 – кнопки управління вікном: згорнути до кнопки на панелі завдань  , розгорнути на весь екран  або згорнути у вікно  , закрити  .

3 – рядок меню вікна (файл, сервіс, вид...);

4 – панелі інструментів, містять кнопки для виконання найбільше часто використовуваних команд. У вікні папки є 3-4 панелі інструментів («*Звичайні кнопки*», «*Адресний рядок*» і т.д.). Вмикання/вимикання панелі інструментів виконується командою **Вид – Панели инструментов – Назва панелі**.

5 – робоча область вікна, у якій відображається вміст відкритого об'єкта.

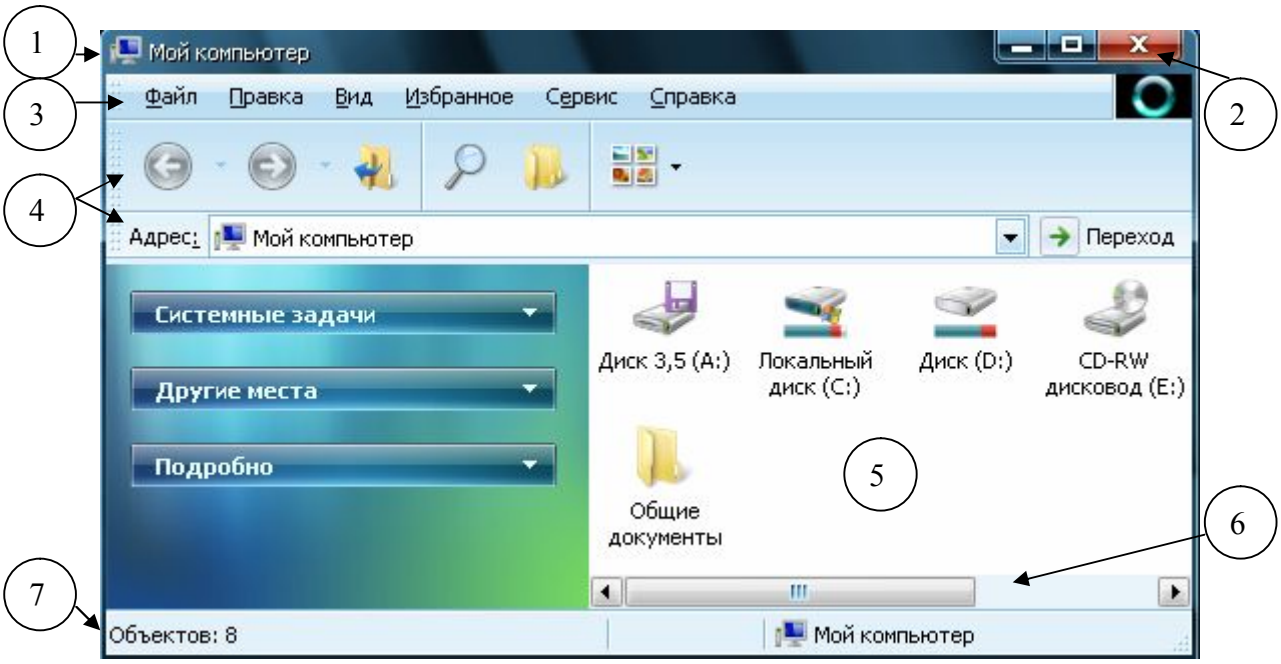



Рисунок 2.7. Структура вікна папки

6 - смуги прокручування. Призначені для прокручування вмісту вікна й відображаються в тому випадку, якщо вміст об'єкта не поміщається повністю у вікні даного розміру.

7 – рядок стану, у якій виводиться довідкова інформація або підказки для користувача. Для її включення використається команда **Вид – Строка состояния**.

### 2.3.4. Робота з вікнами папок

Для переміщення вікна по робочому столі необхідно встановити курсор миші на заголовок вікна, нажати ліву кнопку миші й утримуючи її, рухати вікно по екрану. Для зміни розміру вікна необхідно встановити курсор миші на одну із границь вікна (курсор повинен прийняти вид ), нажати ліву кнопку миші й утримуючи її, пересунути границю вікна.

ОС Windows дозволяє одночасно відкривати на «Робочому столі» кілька вікон. При цьому на панелі задач з'являється відповідна кожному вікну кнопка. Існує два способи огляду папок:

1) *одновіконний режим* – папки відкриваються в тому самому вікні;

2) *багатовіконний режим* – кожна папка відкривається в окремому вікні.

Для вибору однієї з відкритих папок необхідно виконати 1 лкм на видимій частині її вікна або на кнопці цієї папки на панелі задач.

Вміст вікна папки може відображатися в декількох режимах:

– таблиця (виводиться повна інформація про об'єкти, що містяться в папці);

– список (у кілька стовпців виводяться тільки імена об'єктів);

– значки та ін.

Вибір режиму виконується командою **Вид – Назва режиму**.

Вміст вікна папки може бути впорядковане по декількох критеріях:

– по імені;

– по типу;

– по розміру й т.д.

Вибір способу сортування, виконується командою **Вид – Упорядочить значки – Критерій сортування**.

## 2.4. Робота з дисками, папками й файлами

В ОС Windows у більшості випадків визначену команду можна виконати чотирма різними способами:

1) за допомогою меню;

2) за допомогою панелі інструментів;

3) за допомогою контекстного меню, що викликається одинарним щигликом правої кнопки миші (1 пкм) і містить команди, які можна виконати в цей момент часу стосовно до поточного об'єкта.

4) за допомогою комбінації клавіш.

### 2.4.1. Виділення об'єктів

Для виділення одного об'єкта - виконати 1 лкм на його значку.

Для одночасного виділення групи підряд розташованих об'єктів – виділити перший, натиснути клавішу **Shift** і, утримуючи її, виділити останній об'єкт.

Для одночасного виділення декількох об'єктів у розкид – виділити перший, нажати клавішу **Ctrl** і, утримуючи її, по черзі виділити інші об'єкти.

Для виділення всіх об'єктів у вікні папки виконати команду **Правка – Выделить все** або нажати комбінацію клавіш **Ctrl + A.**

### 2.4.2. Створення папки

Відкрити вікно папки, у якій необхідно створити нову папку. Виконати команду **Файл – Создать - Папку**, біля знака нової папки ввести її ім'я й натиснути клавішу **Enter**.

### 2.4.3. Створення файлів

Існує два способи створення файлу користувача.

1 спосіб. *Створення порожнього файлу*. У папці, де необхідно створити новий файл виконати команду **Файл – Создать – Тип файлу**. Як тип файлу

найбільше часто пропонуються наступні типи:

- Текстовий документ;
- Точковий малюнок;
- Документ Microsoft Word;
- Лист Microsoft Excel і т.д.

Біля значка нового файлу ввести його ім'я (якщо не підходить ім'я яке запропоноване за замовчуванням) і натиснути клавішу **Enter**.

Примітка. Якщо ім'я файлу відображається разом з розширенням, то при вводі нового імені, розширення видаляти ***не можна***, тому що це призведе до зміни типу файлу.

2 спосіб. *Створення файлу за допомогою редактора.* Відкрити необхідну програму-редактор (текстовий редактор, графічний редактор та ін.). Ввести вміст файлу (текст, малюнок) і для збереження файлу на диску виконати команду **Файл – Сохранить**. У вікні команди (рис. 2.8) зробити наступні дії:

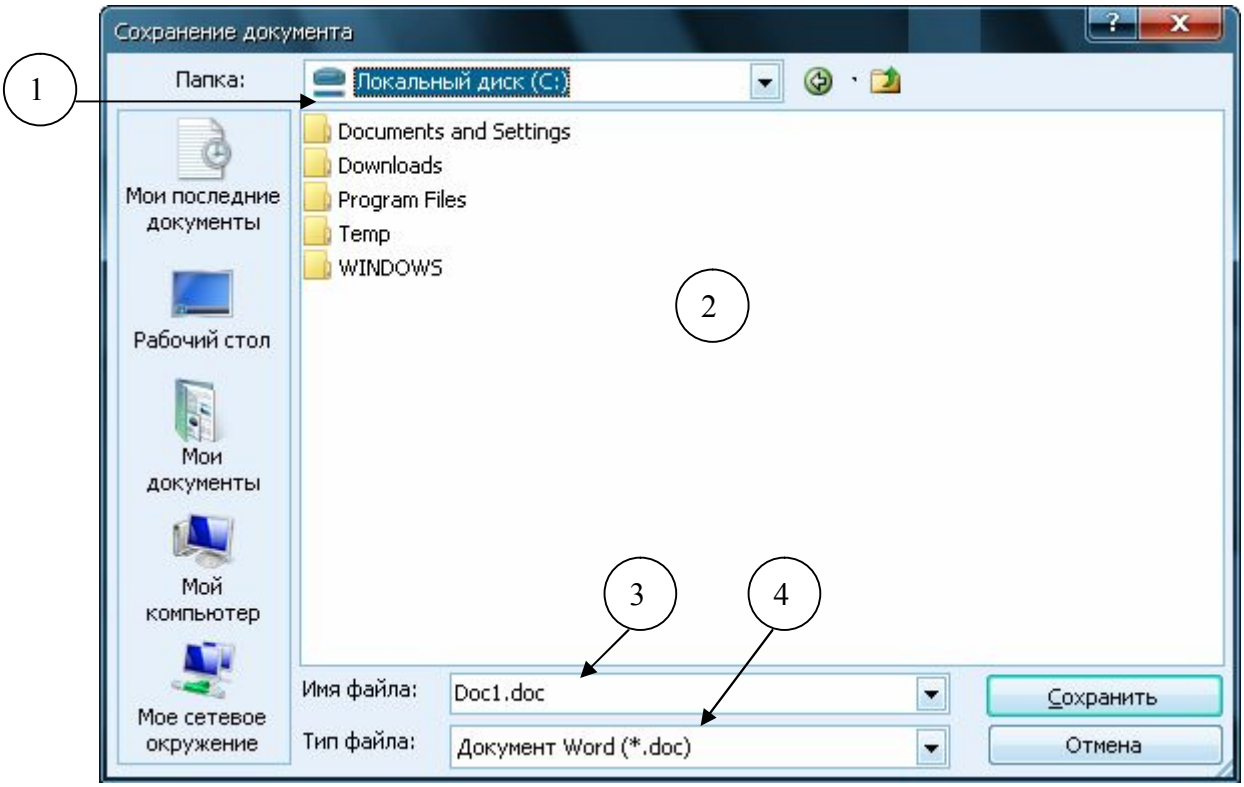



Рисунок 2.8. Вікно «Зберігання документу»

1) У полі «Папка» (1) вибрати зі списку диск, на якому необхідно зберегти файл. В області (2) з'явиться вміст обраного диска.

2) В області (2) знайти й відкрити папку, у якій буде зберігатися файл. Ім'я цієї папки повинне з'явитися в полі (1) .

3) У полі «Ім'я файлу» (3) ввести ім'я створюваного файлу без розширен-

ня. У полі «Тип файлу» (4) вибрати зі списку тип створюваного файлу, якщо не підходить запропонований за замовчуванням. Натиснути кнопку «**Сохранить**».

2.4.4. Перейменування файлів і папок

Виділити об'єкт для перейменування, виконати команду **Файл – Переименовать**. Ввести нове ім'я (див. примітка з п. 2.4.3) і натиснути клавішу **Enter**.

2.4.5. Перегляд властивостей об'єктів

Основними властивостями об'єктів, є тип, розмір, розташування (адреса), дата й час створення. Крім цього файли, папки і ярлики мають набір додаткових засобів – ***атрибутів***: прихований, системний, тільки для читання, архівний. Для перегляду або зміни властивостей виділити об'єкт і виконати команду **Файл – Свойства**.

2.4.6. Копіювання й переміщення об'єктів

Існують кілька основних способів копіювання або переміщення файлів і папок:

1 спосіб. *За допомогою буфера обміну*. ***Буфер обміну*** – це частина оперативної пам'яті ПК, що використовується для тимчасового зберігання даних які будуть копіюватися або переміщуватися.

Для копіювання або переміщення об'єктів виконати наступну послідовність дій:

1) Виділити необхідні об'єкти.

2) Виконати команду **Правка – Копировать** (для наступного копіювання об'єктів) або команду **Правка – Вырезать** (для наступного переміщення об'єктів). Тим самим виділені об'єкти будуть поміщені в буфер обміну.

3) Відкрити вікно папки, у яку необхідно копіювати або переміщати об'єкти. Виконати команду **Правка – Вставить**. Об'єкти, які поміщені в буфер обміну, можна вставляти кілька разів у різні папки.

Для швидкого виконання команд роботи з буфером обміну можна використовувати наступні комбінації клавіш:

Ctrl+З - копіювати;

Ctrl+X - вирізати;

Ctrl+V - вставити.

2 спосіб. *Шляхом перетаскування правою кнопкою миші*. Перед виконанням операції необхідно розташувати на робочому столі вікна папок приймача й

джерела так, щоб вони повністю не перекривали один одного (*джерело* – це папка, що містить об'єкти для копіювання або переміщення; *приймач* – це папка, у яку будуть копіюватися або переміщатися об'єкти). Потім виконати наступну послідовність дій:

1) У вікні папки джерела виділити об'єкти для копіювання або переміщення.

2) Натиснути на виділених об'єктах праву кнопку миші (**ПКМ**) і, утримуючи її, перетягнути об'єкти у вікно папки приймача, де відпустити ПКМ.

3) З контекстного меню (рис.2.9) вибрати необхідну команду.

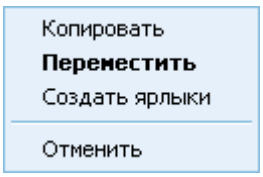



Рисунок 2.9. Контекстне меню, яке з'являється при перетаскуванні ПКМ

Вікно папки приймача можна попередньо не відкривати, досить перетягнути виділені об'єкти на значок цієї папки.

3 спосіб. *Шляхом перетаскування лівою кнопкою миші*. Виконується аналогічно перетаскуванню ПКМ, але в результаті відбувається заздалегідь визначена дія. У випадку якщо папки джерело й приймач розташовані на одному диску, то завжди виконується переміщення об'єктів. Якщо на різних дисках, то завжди виконується копіювання об'єктів. Якщо при перетаскуванні ЛКМ утримувати клавішу **Ctrl**, то завжди буде відбуватися копіювання об'єкта.

### 2.4.7. Видалення об'єктів

Для видалення файлу, папки або ярлика виділити необхідний об'єкт і виконати команду **Файл – Удалить**. При цьому файли не відразу ж віддаляються з диска. На початку вони містяться в системну папку «Кошик» для тимчасового зберігання. Поміщені в кошик файли, можна відновити або остаточно видалити з диска. Для цього необхідно відкрити на робочому столі папку «Кошик», знайти й виділити необхідний файл, виконати команду **Файл – Восстановить** (або **Файл – Удалить**).

Для швидкого видалення файлів з диска, без приміщення їх в «Кошик», натиснути комбінацію клавіш **Shift + Delete**.

### 2.4.8. Створення ярликів

Існують кілька основних способів створення ярликів, застосовуваних у

різних ситуаціях.

1 спосіб. Виділити об'єкт, для якого необхідно створити ярлик, перетягнути його за допомогою ПКМ у папку, де необхідно створити ярлик, з контекстного меню (рис. 2.9) вибрати команду «**Создать ярлыки**». Цей спосіб використається при створенні ярликів для дисків, папок і файлів. Якщо місце розташування файлу не відомо, то на початку необхідно знайти цей файл (команда **Пуск – Найти**), а потім створити для нього ярлик зазначеним способом.

2 спосіб. Виділити об'єкт, для якого необхідно створити ярлик, виконати команду **Правка – Копировать**, відкрити папку, де необхідно створити ярлик, виконати команду **Правка – Вставить ярлык**. Цей спосіб використовується при створенні ярликів для папок і файлів.

3 спосіб. У вікні папки, де необхідно створити ярлик, виконати команду **Файл – Создать – Ярлык**, що запустить майстер створення ярлика.

***Майстер*** – це послідовність діалогових вікон, у яких необхідно включати певні режими, задавати параметри, уводити дані й т.д. для автоматизації процесу одержання кінцевого результату. Кожне діалогове вікно майстра називається ***крок***.

На першому кроці майстра необхідно вказати повний шлях до об'єкта, для якого створюється ярлик (рис. 2.10). Для цього натиснути кнопку «**Обзор**», у вікні команди знайти й відкрити необхідний файл, натиснути кнопку **«Далее»**.

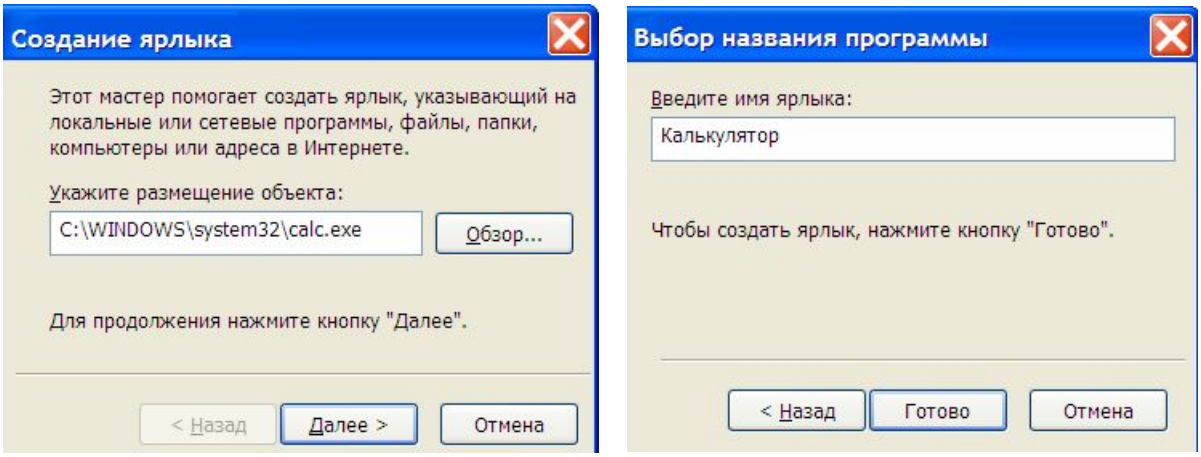



Рисунок 2.10. Майстер створення ярлика

На другому кроці майстра необхідно ввести ім'я ярлика й нажати кнопку «**Готово**». Цей спосіб звичайно використають при створенні ярлика для файлів-програм.

### 2.4.9. Пошук файлів і папок.

Для пошуку файлів на дисках виконати команду **Пуск – Найти – Файлы и папки**. У вікні команди (рис. 2.11) виконати наступні дії:

1) у полі «Частина імені файлу...» (1) увести ім'я шуканого файлу або шаблон, якому воно повинне задовольняти;

2) у полі «Пошук в:» (2) вибрати зі списку диск на якому буде виконуватися пошук файлів;

3) як додаткові критерії пошуку можна задати наступні параметри файлу:

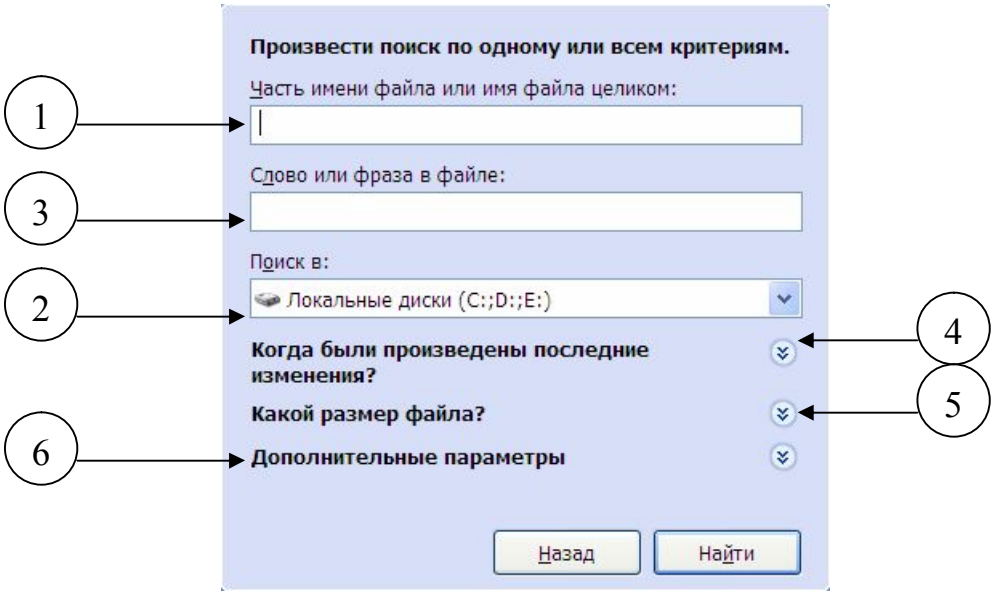



Рисунок 2.11. Вікно пошуку файлів

а) у полі «Слово або...» (3) увести фрагмент тексту, що повинен містити шуканий файл;

б) для завдання обмежень на дату зміни або створення файлу відкрити область «Коли були зроблені останні зміни?» (4), натиснувши кнопку ;

в) для завдання обмеження на розмір шуканого файлу в Кбайтах, відкрити область «Який розмір файлу?» (5);

г) для вибору типу шуканого файлу відкрити область «Додаткові параметри» (6).

Після завдання всіх критеріїв пошуку, нажати кнопку «**Найти**». У правій частині вікна буде показаний список всіх знайдених файлів, що задовольняють критеріям пошуку, із зазначенням їхньої адреси.

## 2.5. Архівація даних

***Архівація*** – це процес стиску вмісту у файлах з метою зменшення їхнього розміру. У результаті архівації створюється файл – ***архів***, що містить у собі вихідні файли в стиснутому виді. Програма, що виконує стиснення файлів називається ***архіватором*** (WinRAR, WinZip і т.д.).

Архіватор WinRAR дозволяє створювати архіви наступних типів:

1) звичайний (суцільний) архів - являє собою один файл із розширенням

RAR;

2) багатотомний архів - це архів, що розділений на частині (том), кожний том - це окремий файл із розширенням R01, R02 і т.д., залежно від кількості томів;

3) архів, що саморозпаковується, - являє собою файл із розширенням EXE, у якому крім самого архіву втримується програма для автоматичного витягу вмісту архіву.

### 2.5.1. Створення архіву

Виділити файли (папки), які необхідно помістити в архів, виконати 1 пкм на виділених файлах, і з контекстного меню вибрати команду «**Добавить в архив**». У вікні «Ім'я й параметри архіву» (рис.2.12) виконати наступні дії:

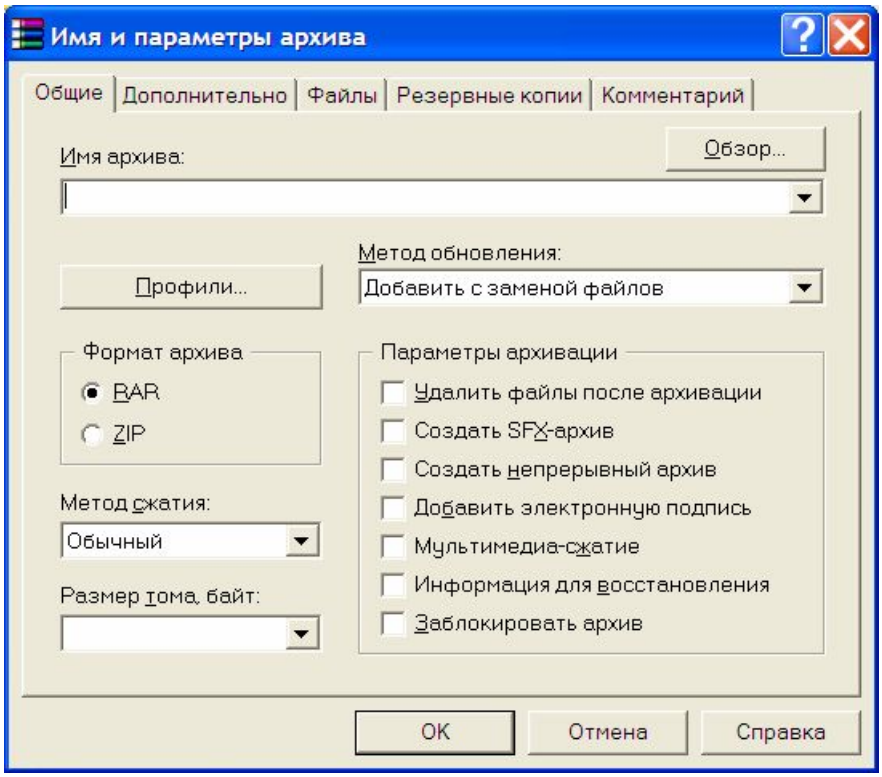



Рисунок 2.12. Створення архіву

1) у поле «Ім'я архіву» увести ім'я створюваного архіву, якщо не підходить ім'я запропоноване за замовчуванням;

2) якщо необхідно створити архів не в поточній папці, то натиснути кнопку «**Обзор**» і у вікні команди знайти й відкрити папку, у яку необхідно помістити архів;

3) для створення багатотомного архіву в поле «Розмір тому, байт:» указати розмір одного тому в байтах;

4) для створення архіву, що саморозпаковується, включити режим «Створити SFX - архів»;

5) для видалення початкових файлів після створення архіву необхідно

включити режим «Видалити файли після архівації».

Для автоматичного створення архіву виконати 1 пкм на виділених файлах, і з контекстного меню вибрати команду «**Добавить в** *ім'я архіву*.**rar**».

Для додавання файлів у вже існуючий архів необхідно виділити їх, виконати 1 пкм на виділених файлах і з контекстного меню вибрати команду «**Добавить в архив**». У вікні команди натиснути кнопку «**Обзор**» і у вікні команди знайти й відкрити архів, у який додаються нові файли.

### 2.5.2. Перегляд архіву

Для відкриття й перегляду вмісту архіву виконати 2 лкм на файлі-архіві. Буде запущена програма WinRAR, у вікні якої відобразиться вміст архіву (список файлів).

### 2.5.3. Витяг файлів з архіву

1 спосіб. *Для звичайних і багатотомних архівів.*

Виконати 1 пкм на файлі-архіві й з контекстного меню вибрати команду «**Извлечь файлы …**». У вікні команди «Шлях і параметри витягу» (рис. 2.13) буде показаний шлях до папки, у яку пропонується витягати файли. Якщо запропонований шлях не підходить, то за допомогою дерева папок відкрити папку, у яку необхідно витягти файли.

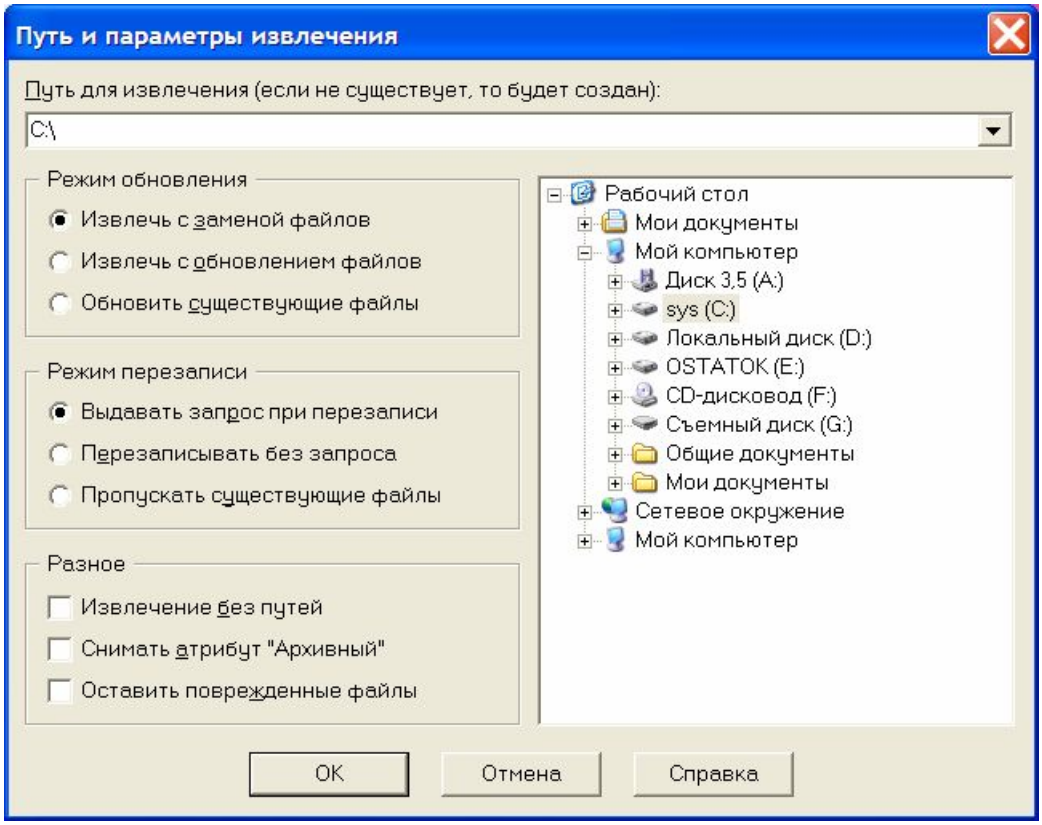



Рисунок 2.13. Витяг файлів з архіву

Для витягу окремих файлів з архіву необхідно відкрити файл-архів для перегляду, знайти й виділити необхідний файл і натиснути кнопку «**Извлечь**». Буде відкрите вікно рис. 2.13.

2 спосіб. *Для архівів, що* саморозпаковуються.

Виконати 2 лкм на файлі-архіві й у вікні, що з'явилося, натиснути кнопку **«Извлечь»**.

# Лекція №3. ТЕКСТОВИЙ РЕДАКТОР MICROSOFT WORD

## 3.1. Загальні теоретичні положення

Найпоширеніший текстовий редактор Microsoft Word входить у пакет офісних додатків Microsoft Office, що є комерційним програмним продуктом. Редактор призначений для створення й обробки текстових документів будь-якого рівня складності й дозволяє використати графічні об'єкти.

Основу додатка Microsoft Word становить файл Winword.exe. Робочий файл редактора називається ***Документ***, має розширення doc і звичайно являє собою одну або кілька сторінок формату А4.

## 3.2. Структура вікна редактора

Структура вікна редактора MS Word (рис. 3.1) всіх версій є стандартної, невеликі відмінності з'являються, починаючи з версії пакета Microsoft Office 2007.

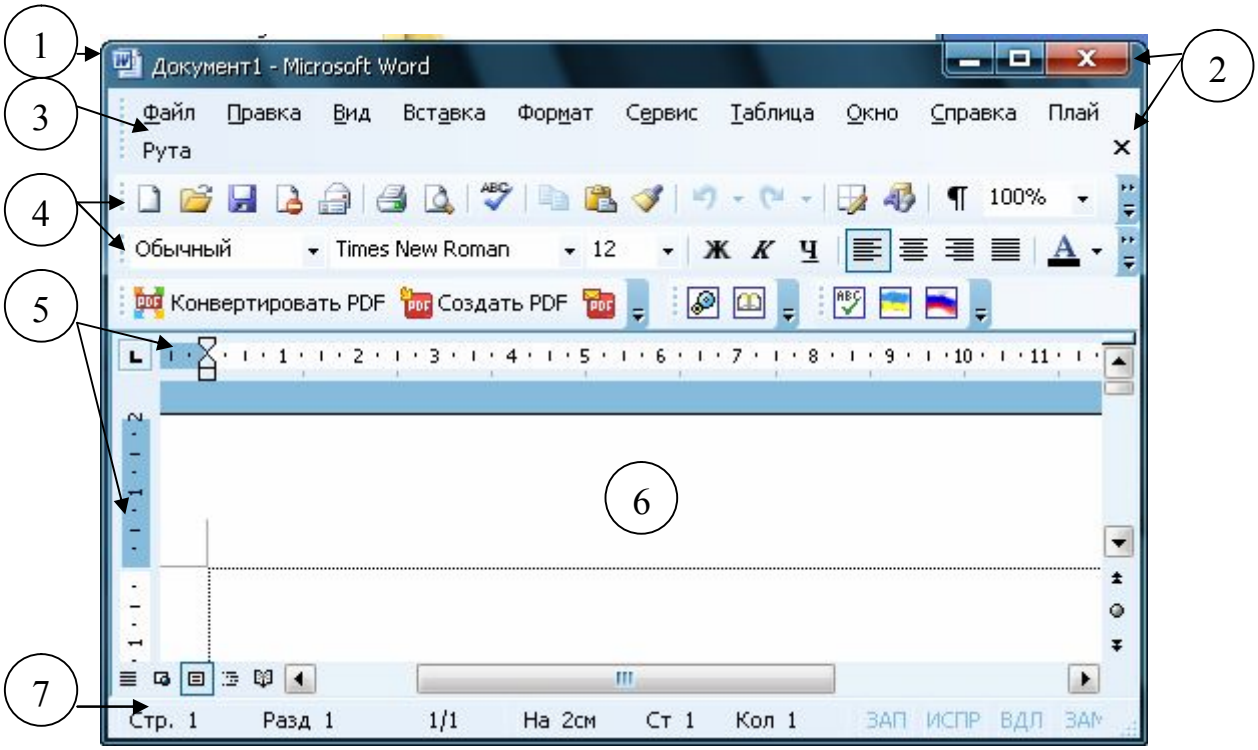



Рисунок 3.1. Вікно редактора MS Word

1 – заголовок вікна редактора, у якому виводиться назва відкритого документа;

2 – кнопки управління вікном редактора (угорі) і вікном документа (унизу);

3 – меню редактора;

4 – панелі інструментів. У редакторі є більш 15 панелей інструментів. При цьому основними є панелі «*Стандартна*» і «*Форматування*». Включення інших панелей інструментів виконується командою **Вид – Панели инструментов – Назва панелі**;

5 – горизонтальна й вертикальна лінійки, що показують розміри сторінки в сантиметрах. У випадку відсутності лінійки, її можна включити командою **Вид – Линейка**;

6 – сторінка документа;

7 – рядок стану.

Документ може відображатися в різному масштабі, що вибирається командою **Вид – Масштаб**, або за допомогою кнопки «**Масштаб**» 100% на панелі інструментів «*Стандартна*».

Працювати з документом можна в різних режимах: *звичайний*, *розмітка сторінки*, *структура* та ін. Вибір режиму виконується командою **Вид – Назва режиму**. Найбільш зручним режимом для роботи з текстом є режим «Розмітка сторінки».

## 3.3. Робота з файлом-документом

### 3.3.1. Створення документа

При відкритті редактора завжди автоматично створюється новий документ. Будь-який документ завжди створюється на основі шаблону. ***Шаблон*** – це файл із розширенням dot, у якому містяться параметри документа. Звичайний документ створюється на основі шаблону Normal.dot, що зберігається в системній папці *Шаблони* (кнопка «**Создать файл (по умолчанию)**» ). Крім цього в редакторі є велика кількість стандартних шаблонів для створення листів, факсів, записок, резюме та ін. Щоб створити новий документ на основі одного з таких шаблонів необхідно виконати команду **Файл – Создать**. На панелі «Створення документа» вибрати режим «Шаблони на моєму комп'ютері». У вікні «Шаблони» (рис. 3.2) вибрати необхідний шаблон, включити режим «Створити документ» і натиснути кнопку Ok.

Щоб створити новий шаблон необхідно у вікні «Шаблони» (рис. 3.2) включити режим «Створити шаблон», задати параметри документа й при необхідності ввести його стандартний уміст. Зберегти отриманий шаблон у папці

*Шаблони.*

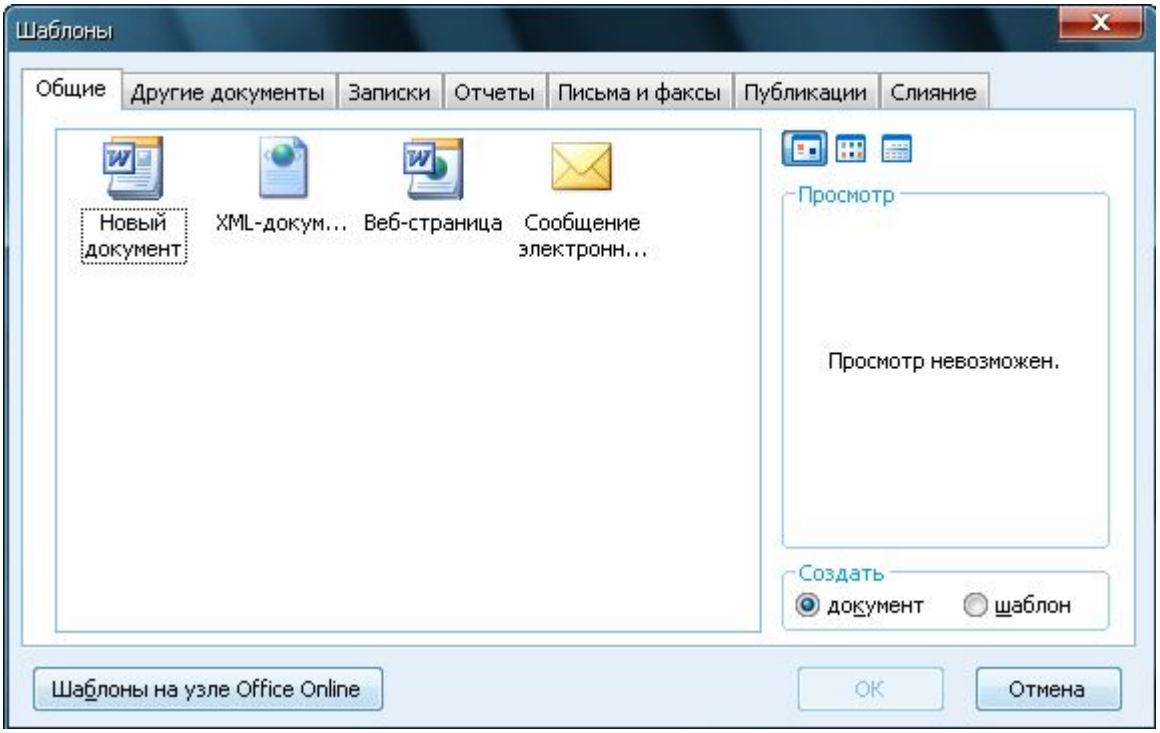



Рисунок 3.2. Використання шаблонів

### 3.3.2. Збереження документа

Існують три основних варіанти збереження документа.

1) Для збереження нового документа виконати команду **Файл – Сохранить** й виконати зазначені в п. 2.4.2 дії.

2) Для збереження змін у раніше створеному документі виконати команду **Файл – Сохранить** або натиснути кнопку «**Сохранить**» .

3) Для збереження раніше створеного файлу під іншим іменем або в іншій папці виконати команду **Файл – Сохранить как** і у вікні команди (рис. 2.8) виконати необхідні дії.

### 3.3.3. Відкриття документа

Найпростіший спосіб відкриття документа Microsoft Word – це виконати 2 лкм на значку файлу у вікні папки. Для відкриття документа з вікна редактора необхідно виконати команду **Файл – Открыть** (кнопка «**Открыть**» ). У вікні команди (рис. 3.3):

1) У полі «Папка» (1) вибрати зі списку диск, на якому знаходиться необхідний файл. В області (2) з'явиться вміст обраного диска.

2) В області (2) знайти й відкрити папку, у якій міститься файл. Ім'я цієї

папки з'явитися в полі (1) .

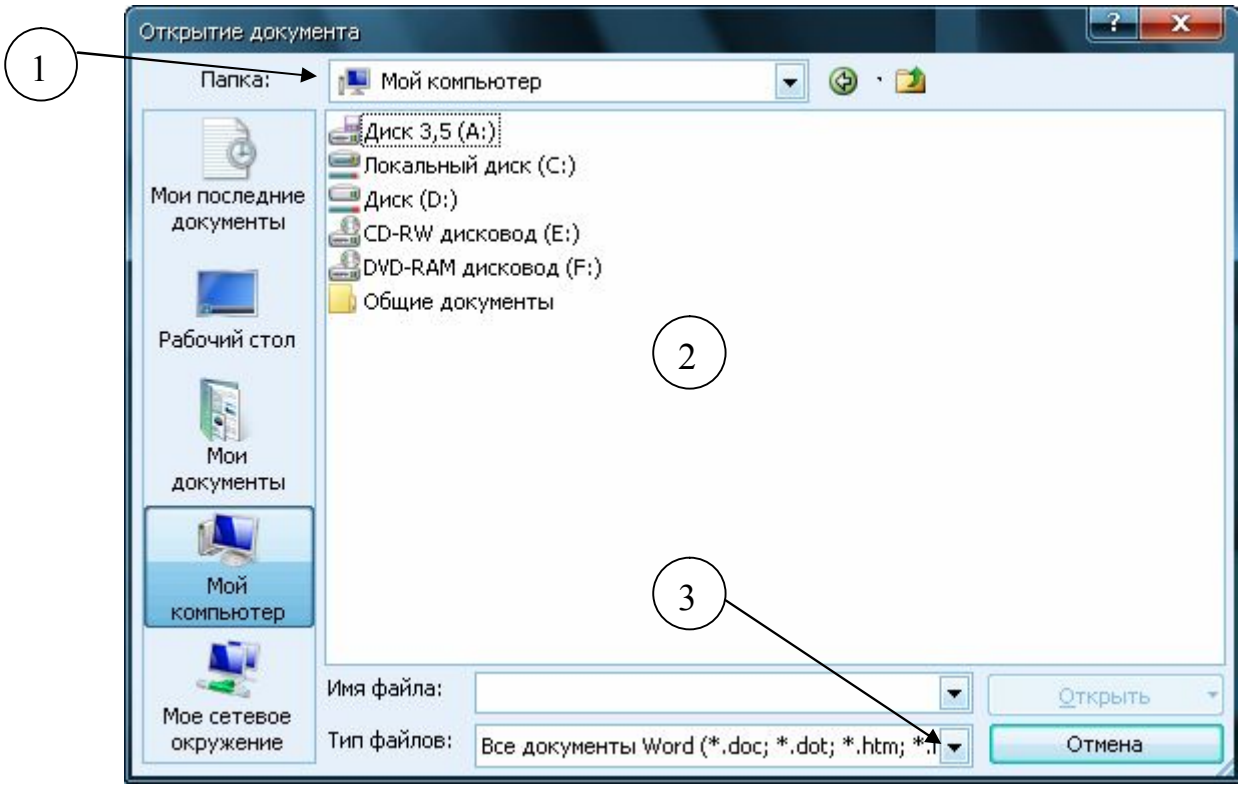



Рисунок 3.3. Вікно «Відкриття документа»

3) В області (2) знайти й виділити необхідний файл, натиснути кнопку «**Открыть**». Якщо необхідно відкрити не стандартний документ редактора, то в поле «Тип файлу» (3) вибрати зі списку режим «**Все файлы *.***». Це дозволить відобразити всі файли, які є в обраній папці.

### 3.3.4. Парламенти сторінок документа

Для зміни параметрів сторінки документа виконати команду **Файл – Параметры страницы** або 2 лкм на лінійці. Вікно команди (рис. 3.4) містить наступні вкладки:

1) **Поля** – призначена для встановлення розмірів полів сторінки, які задаються в сантиметрах (рис. 3.5). Прийнято наступні стандартні розміри полів: ліве поле – 3-2,5 см; праве поле – 1-1,5 см; верхнього й нижнього поля по 2 см. Також є можливість вибирати спосіб орієнтації сторінок у документі: книжкова або альбомна. При цьому можна змінити орієнтацію як всіх сторінок у документі (у полі «**Застосувати**:» вибрати режим «*до всього документа*»), так і окремих виділених сторінок (у полі «**Застосувати**:» вибрати режим «*до виділеного тексту*»).

2) **Розмір паперу** – призначена для встановлення розміру сторінки шляхом вибору її формату (A4, A5, Letter та ін.) або зазначення її ширини й висоти

в сантиметрах.

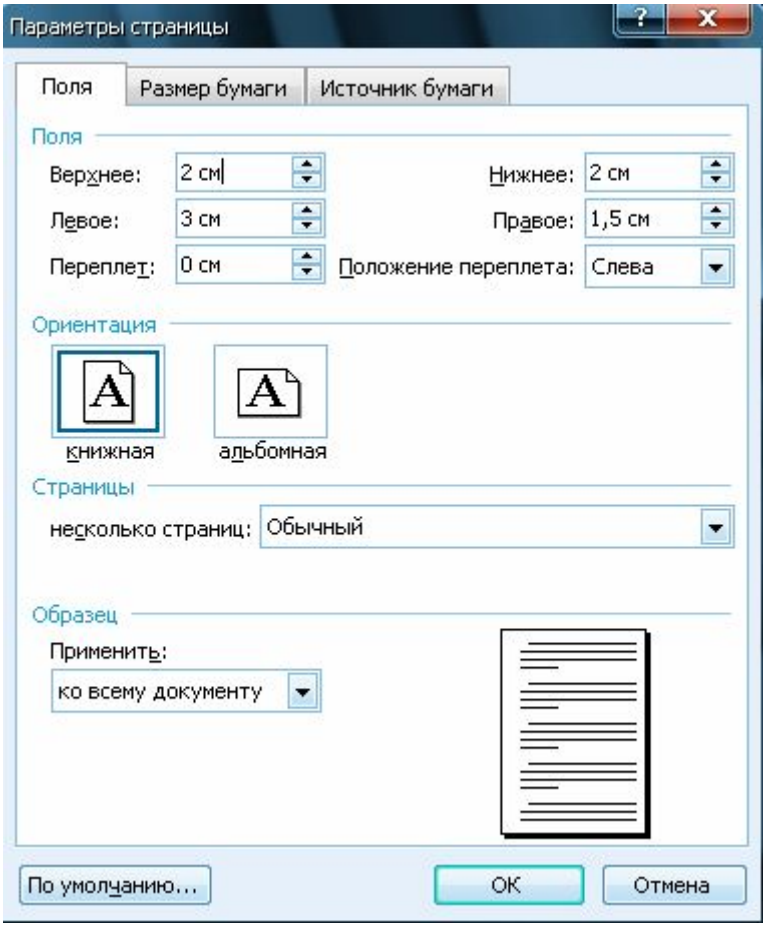



Рисунок 3.4. Вікно «Параметри сторінки»

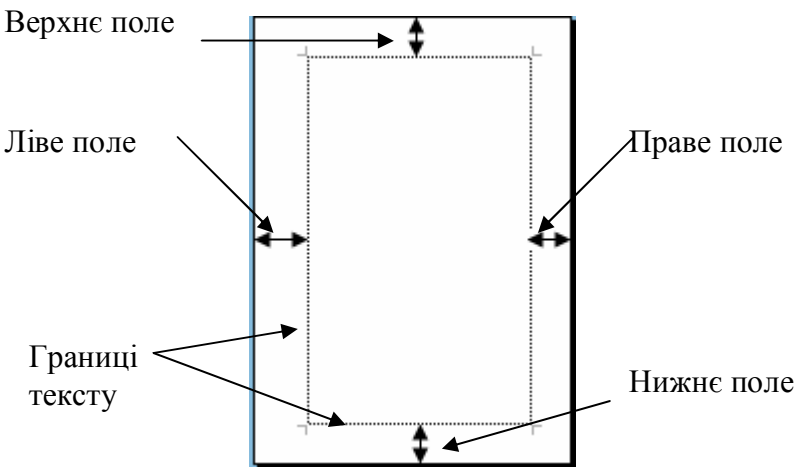



Рисунок 3.5. Поля на сторінці

Якщо границі тексту (рис. 3.5) не відображаються на сторінці, то виконати команду **Сервис – Параметры**, вибрати вкладку **Вид** і включити режим «*Границі тексту*».

## 3.4. Робота з текстом

### 3.4.1. Виділення фрагментів тексту

Для виділення довільного фрагмента тексту встановити курсор миші в

початок фрагмента, натиснути ЛКМ і, утримуючи її, розтягти область виділення на весь фрагмент. Для виділення фрагмента тексту за допомогою клавіатури встановити курсор, що показує позицію вводу (**КППВ**) у початок фрагмента, натиснути клавішу **Shift** і, утримуючи її, за допомогою стрілок на клавіатурі, виділити необхідний фрагмент. При одночасному виділенні декількох окремих фрагментів тексту необхідно натиснути й утримувати клавішу **Ctrl**.

Також існують різні способи прискореного виділення фрагментів тексту:

1) для виділення слова виконати 2 лкм на ньому;

2) для виділення речення натиснути клавішу **Ctrl** і, утримуючи її, виконати 1 лкм на будь-якій частині речення;

3) для виділення рядка тексту виконати 1 лкм в області лівого поля біля рядка (курсор миші повинен мати вигляд  );

4) для виділення декількох рядків тексту натиснути 1 лкм в області лівого поля біля 1-й рядка й рухати курсор миші (курсор миші повинен мати вигляд  ) нагору або вниз по полю;

5) для виділення абзацу в тексті виконати 2 лкм в області лівого поля біля абзацу (курсор миші повинен мати вигляд  ) або 3 лкм у будь-якій частині абзацу.

### 3.4.2. Копіювання й переміщення фрагментів тексту

Копіювання й переміщення фрагментів тексту найбільше зручно виконувати за допомогою буфера обміну (п. 2.4.6). Для цього необхідно:

1) виділити необхідний фрагмент тексту;

2) виконати команду **Правка – Копировать** для наступного копіювання фрагмента (кнопка «**Копировать**»  ) або команду **Правка – Вырезать** для наступного переміщення фрагмента (кнопка «**Вырезать**»  );

3) установити курсор КППВ у необхідну позицію й виконати команду **Правка – Вставить** (кнопка «**Вставить**»  ).

Також для копіювання або переміщення фрагмента тексту можна використати перетаскування ПКМ.

### 3.4.3. Правила введення тексту

При введенні тексту бажано дотримуватися наступних простих правил:

1) Якщо при введенні тексту курсор КППВ досяг границі правого поля, то **натискати клавішу Enter** для переходу на наступний рядок **не треба**. Курсор автоматично перейде на наступний рядок при продовженні введення.

2) Абзацний відступ за допомогою пробілів або клавіші **Tab** не робити.

3) Перед розділовими знаками пробіли не ставити, після - ставити обов'я-

зково.

У новому документі частина сторінки, розташована нижче курсору КППВ є недоступною для введення тексту. Для переведення курсору в цю область необхідно виконати 2 лкм у необхідній позиції або натискати клавішу **Enter** поки курсор не переміститься туди.

## 3.5. Форматування тексту

Під форматуванням розуміється зміна зовнішнього вигляду й параметрів основних елементів тексту: ***символів (шрифт)*** і ***абзаців***. Форматування можна виконувати як до початку введення тексту, так і після. У другому випадку фрагмент тексту необхідно попередньо виділити.

### 3.5.1. Форматування шрифту

Виділити необхідний фрагмент тексту й виконати команду **Формат – Шрифт**. Вікно команди (рис. 3.6) містить наступні вкладки:

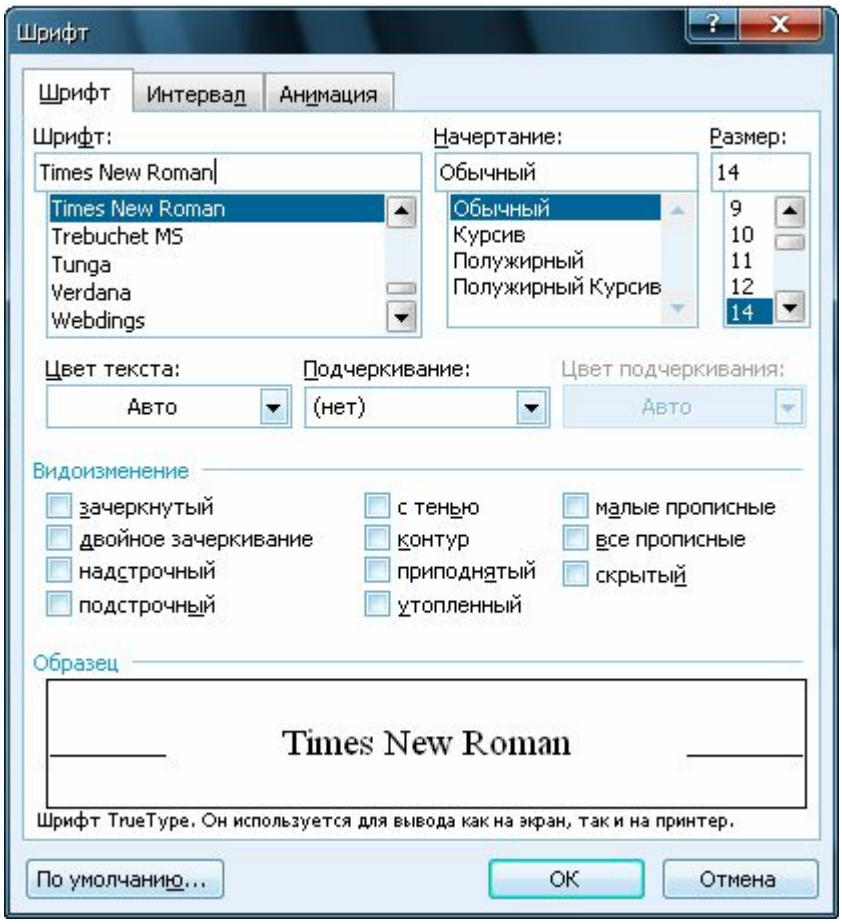



Рисунок 3.6. Вікно «Шрифт». Вкладка **Шрифт**

**Шрифт** – вкладка призначена для зміни таких параметрів шрифту як:
а) *тип шрифту*. Частина шрифтів (кнопка «**Шрифт**» 

),

установлених в ОС Windows не підтримують відображення літер кирилиці, тому при виборі типу шрифту необхідно звертати увагу на пропонований зразок. Стандартним для оформлення документів вважається шрифт **Times New Roman**.

б) *розмір шрифту*, що задається в пунктах **пт** (1 пт = 1/72 дюйма ≈ 0,35 мм). При оформленні документів звичайно використовують шрифт розміром 14 пт (кнопка «**Выбрать размер шрифта**» 14 ).

в) *спосіб написання*: звичайний, **напівжирний** (кнопка «**Полужирный**» Ж), *курсив* (кнопка «**Курсив**» К ), ***напівжирний курсив***.

г) *спосіб підкреслення*: суцільне (кнопка «**Подчеркнутый**» Ч), тільки слова та ін. При обраному способі підкреслення можна додатково задавати *цвіт підкреслення*.

д) *цвіт тексту* (кнопка «**Цвет шрифта**» A).

е) *видозміни (ефекти)*: ~~закресленний~~, контур, з тінню та ін. При цьому найбільш корисними є такі ефекти як надрядковий (кнопка «**Надстрочный**» $x^2$) і підрядковий (кнопка «**Подстрочный**» $x_2$) символи.

**Інтервал** – вкладка (рис. 3.7) призначена для зміни таких параметрів шрифту як:

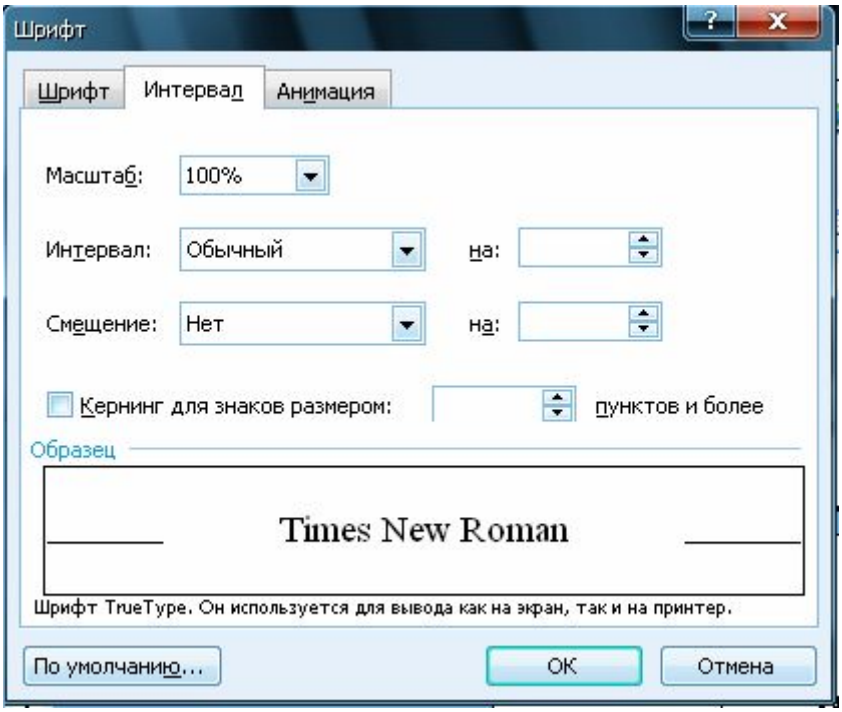



Рисунок 3.7. Вікно «Шрифт». Вкладка **Інтервал**

а) *масштаб відображення символів* у тексті.

б) *межсимвольний інтервал*: звичайний, розріджений, ущільнений. Інтервал задає відстань між символами в пунктах.

в) *зсув*, що зрушує нагору або вниз на задану кількість пунктів символи в рядку.

г) *кернінг для знаків розміром N пунктів і більше*. Кернінг - це виборча зміна інтервалу між літерами залежно від їхньої форми.

**Анімація** – вкладка дозволяє оформляти текст різними анімаційними ефектами: мерехтіння, феєрверк та ін.

### 3.5.2. Форматування абзацу

Для форматування одного абзацу його можна не виділяти, досить поставити в абзац КППВ. Для одночасного форматування декількох абзаців необхідно виділити їх і виконати команду **Формат – Абзац**. Вікно команди (рис. 3.8) містить дві вкладки:

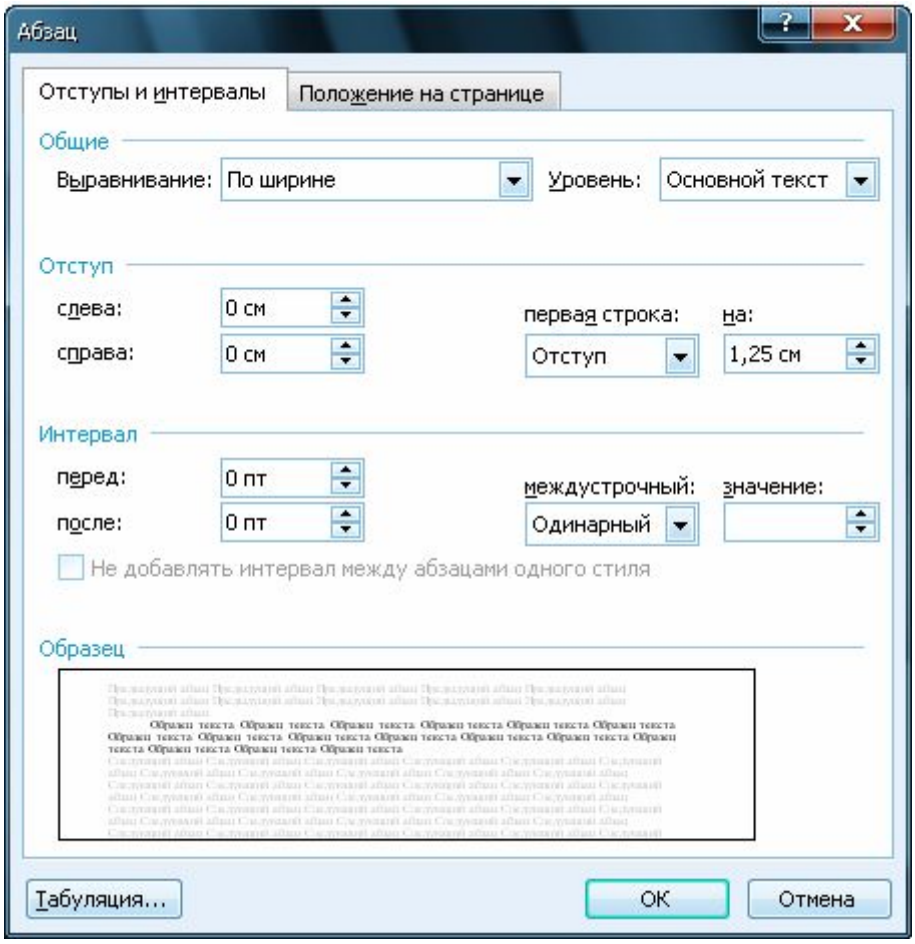



Рисунок 3.8. Вікно «Абзац». Вкладка **Відступи й інтервали**

**Відступи й інтервали** – вкладка призначена для зміни таких параметрів абзацу як:

а) *спосіб вирівнювання* тексту в абзаці. Існують наступні способи вирівнювання тексту в абзаці (рис. 3.9):

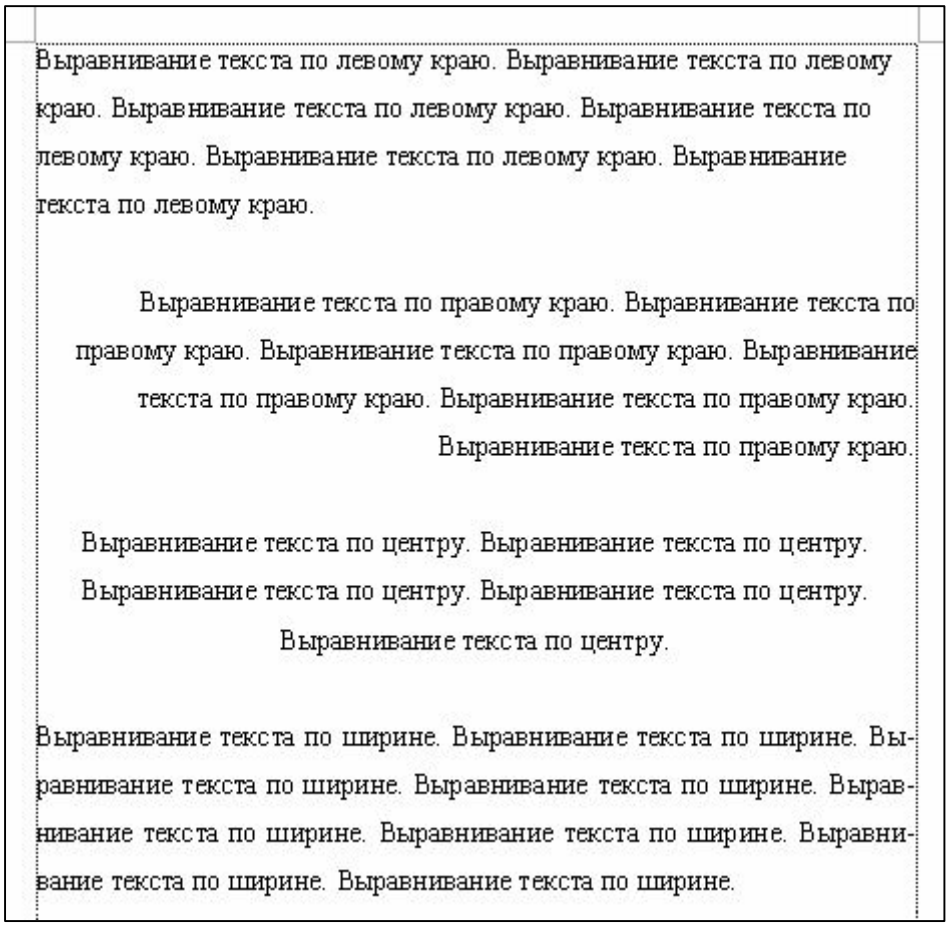



Рисунок 3.9. Способи вирівнювання тексту в абзаці.

по ***лівому краю*** – всі рядки абзацу починаються на одному рівні (кнопка «**По левому краю**» ), тобто лівий край абзацу рівний;

по ***правому краю*** – всі рядки абзацу закінчуються на одному рівні (кнопка «**По правому краю**» ), тобто правий край абзацу рівний;

по ***центрі*** – всі рядки абзацу центруються щодо границь тексту (кнопка «**По центру**» );

по ***ширині*** – всі рядки абзацу починаються й закінчуються на одному рівні (кнопка «**По ширине**» ), тобто обидва краю абзацу рівні.

При форматуванні тексту звичайно використовується вирівнювання по ширині, заголовки в тексті вирівнюються по центру.

б) *відступи* границь абзацу *ліворуч* і *праворуч* від границь полів. Розмір відступу вказується в сантиметрах. Розмір відступу (рис. 3.10) може бути заданий як додатним числом (границі абзацу виходять на поля), так і відємним числом (границі абзацу зміщаються до центра сторінки).

Для того щоб текст в абзаці починався й закінчувався на границях полів відступи ліворуч і праворуч повинні бути рівні 0.


Рисунок 3.10. Відступи абзацу ліворуч і праворуч

в) *відступ першого рядка* абзацу на задану кількість сантиметрів може бути указаний у вигляді *відступу* або *виступу*. Таким чином, задається «новий рядок» в абзаці (рис. 3.11).

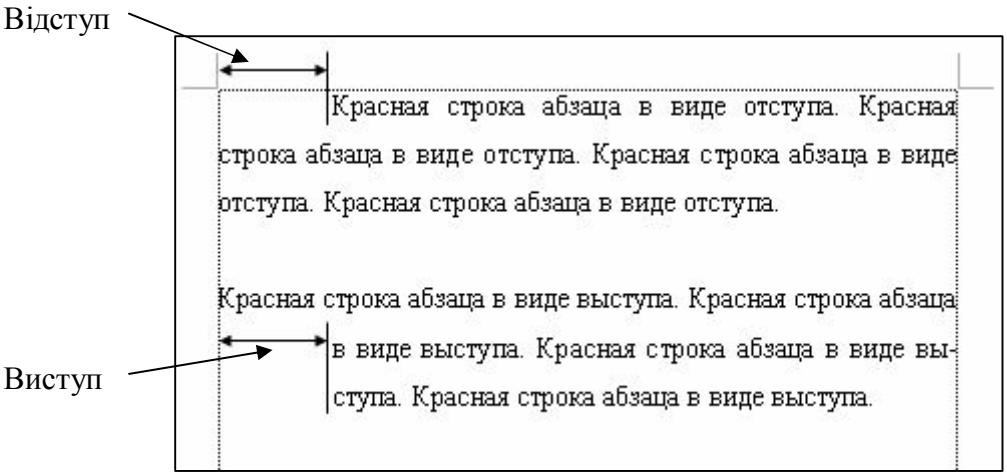



Рисунок 3.11. Відступи першого рядка абзацу

г) *інтервал перед і після* абзацу вказується в пунктах і задає відстань між абзацами.

д) *інтервал міжрядковий* указується в пунктах і задає відстань між суміжними рядками в абзаці. Міжрядковий інтервал може приймати наступні значення: одинарний, подвійний, множник та ін. При оформленні документів найбільше часто використовується полуторний міжрядковий інтервал.

**Положення на сторінці** – вкладка призначена для завдання наступних

параметрів розбивки сторінки й абзацу:

а) *заборона висячих рядків* – цей параметр не дозволяє розривати абзац на границі двох сторінок, якщо при цьому на одній зі сторінок буде залишений «висяча» рядок. Перший або останній рядок абзацу називається висячій, якщо вона розташовується в самому кінці/початку сторінки. У більшості випадків рекомендується цей режим відключати.

б) *не розривати абзац* – цей параметр не дозволяє розривати абзац на границі двох сусідніх сторінок.

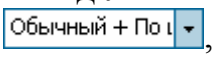
в) *не відривати від наступного* – цей параметр не дозволяє розривати сторінку між поточним абзацом і абзацом, що іде за ним.

г) *з нової сторінки* – цей параметр вставляє перед абзацом примусовий розрив сторінки.

д) *заборонити нумерацію рядків* – цей параметр не дозволяє відображати при печатці номера рядків в абзаці.

е) *заборонити автоматичний перенос слів* – цей параметр не дозволяє переносити слова в межах абзацу, якщо в документі включений режим автоматичного переносу слів (п. 3.7).

### 3.5.3. Стиль форматування

Стиль – це сукупність параметрів форматування шрифту й абзацу. Кожен стиль форматування має власне ім'я. Для використання одного з наявних стилів необхідно виділити фрагмент тексту, потім за допомогою кнопки «**Стиль**» 

, вибрати зі списку потрібний варіант стилю.

Для створення нового стилю виконати команду **Формат – Стили и форматирование**, на додатковій панелі натиснути кнопку «**Создать стиль**», задати параметри форматування шрифту й абзацу й указати ім'я нового стилю.

Для копіювання наявного стилю оформлення виділити початковий фрагмент тексту з потрібним стилем, на панелі інструментів нажати кнопку «**Копировать формат**»  , потім курсором миші у вигляді «пензлика» виділити фрагмент тексту, але який копіюється стиль форматування.

### 3.5.4. Використання лінійки для форматування абзацу

***Лінійка*** (рис. 3.12) не тільки показує розмір сторінки документа, але й дозволяє встановлювати розміри полів сторінки (горизонтальна й вертикальна лінійки), а також установлювати абзацні відступи й позиції табуляції (горизонтальна лінійка).

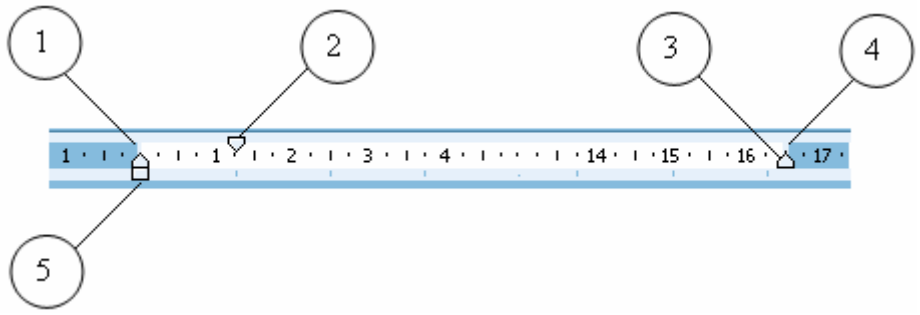



Рисунок 3.12. Використання горизонтальної лінійки

1 - зміна розміру лівого поля сторінки;
2 - маркер зміни абзацного відступу;
3 - маркер зміни відступу праворуч в абзаці;
4 - зміна розміру правого поля сторінки;
5 - маркер зміни відступу ліворуч в абзаці.

### 3.5.5. Додаткові команди форматування тексту

Команда **Формат – Регистр** дозволяє в попередньо виділеному фрагменті тексту перетворити всі символи в *прописні*, *рядкові* й т.д.

Команда **Формат – Колонки** дозволяє розбити попередньо виділений фрагмент тексту на окремі колонки.

Команда **Формат – Границы и заливка** дозволяє задати границі (обрамлення) для елементів тексту й документа. Вікно команди містить три вкладки:

**Границя** – вкладка дає можливість задавати границі для окремих рядків або абзаців тексту. При виконанні команди необхідно вибрати тип границі; тип лінії, її кольори й ширину. Якщо в поле «**Применить к**» вибрати режим «**абзацу**», то в попередньо виділеному фрагменті тексту границя буде задана для абзацу цілком. Якщо необхідно задати границю для окремого рядка або фрагмента тексту, то в поле «**Применить к**» вибрати режим «**тексту**».

**Сторінка** – вкладка дає можливість задавати границі для цілої сторінки.

**Заливання** – вкладка призначена для вибору кольорів заливання обмеженої границями області.

Команда **Формат – Фон** дозволяє вибрати спосіб і кольори заливання всіх сторінок документа.

## 3.6. Робота зі списками

### 3.6.1. Створення однорівневих списків

Редактор дозволяє створити списки наступних типів: *маркірований*, *нумерований* і *багаторівневий*, котрий також ділиться на маркірований і нумерова-

ний списки. Кожний пункт списку являє собою окремий абзац тексту. Для створення списку необхідно виконати команду **Формат – Список**.

У вікні команди (рис. 3.13) потрібно вибрати вкладку з відповідним типом списку, потім вибрати вид списку. Для зміни таких параметрів списку як положення номера (маркера), формат номера (знак маркера) і положення тексту (відступ) необхідно натиснути кнопку «**Изменить**». Якщо в документі вище по тексту є аналогічний список, то для нового списку вказати чи буде він продовжувати попередній список (режим «продовжити») або починатися заново (режим «почати заново»). Для відновлення параметрів обраного списку, які задані за замовчуванням – натиснути кнопку «**Сброс**».

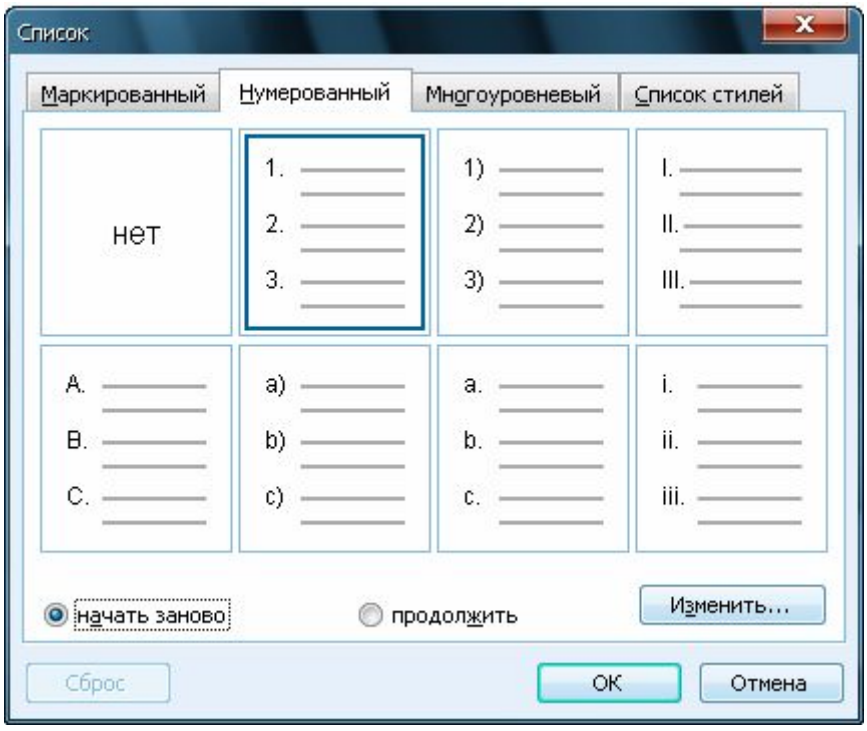



Рисунок 3.13. Вікно «Список»

Після натискання кнопки «**Ок**» у документ буде доданий перший пункт списку. Для додавання наступних пунктів потрібно натискати клавішу **Enter**. Для видалення останнього «зайвого» пункту списку натиснути клавішу **Backspace**.

Для перетворення звичайного тексту в список або зміни параметрів наявного списку, необхідно виділити відповідний фрагмент, виконати команду **Формат – Список** і зробити зазначені вище дії. Для перетворення списку у звичайний текст необхідно виділити його й у вікні команди «Список» вибрати режим «**Ні**».

Для швидкого створення списку можна використовувати кнопки:

нумерований список за замовчуванням;

маркірований список за замовчуванням.

3.6.2. Створення багаторівневих списків

Багаторівневий список (рис. 3.14) дозволяє організувати ієрархічну структуру нумерації в документі. Такий список може містити до дев'яти рівнів ієрархії.


Нумерований Маркірований

Рисунок 3.14. Приклади багаторівневих списків

Багаторівневий список створюється аналогічно одноріневому. При цьому в документ буде доданий перший пункт першого рівня, для додавання наступних пунктів поточного рівня потрібно натискати клавішу **Enter**. Для переходу на наступний рівень списку необхідно натиснути клавішу **Tab** або кнопку «**Увеличить отступ**»  . Для повернення на попередній рівень списку необхідно натиснути комбінацію клавіш **Shift+Tab** або кнопку «**Уменьшить отступ**»  .

## 3.7. Розміщення переносів у документі

Редактор дозволяє виконати автоматичне розміщення переносів у всьому тексті документа. Для цього необхідно виконати команду **Сервис – Язык – Расстановка переносов**. У вікні команди включити режим «*Автоматичне розміщення переносів*».

У заголовках документа переносів бути не повинне, для них необхідно включити режим «*заборонити автоматичний перенос слів*» (п. 3.5.2).

## 3.8. Перевірка правопису в документі

Редактор автоматично підкреслює фрагменти тексту в наступних випадках:

1) *червоною хвилястою лінією* – це значить, що слово містить граматичні помилки або відсутній у словнику редактора. Для виправлення помилки виконати 1 пкм на слові, і з контекстного меню вибрати:

а) правильний варіант написання слова (якщо він є);

б) команду «**Пропустить**» для виключення слова із процесу перевірки

правопису;

в) команду «**Добавить в словарь**» для додавання слова в словник редактора, якщо воно не містить помилок.

2) *зеленою хвилястою лінією* – це значить, що фрагмент тексту містить синтаксичні помилки (на правильно розставлені розділові знаки, немає зв'язку слів у пропозиції, неправильна структура пропозиції та ін.). Для перегляду причини помилки і її виправлень виконати 1 пкм на цьому фрагменті тексту.

3) *червоною хвилястою лінією* – але фрагмент тексту не містить граматичних помилок. Це значить, що не правильно обрано мову для перевірки правопису. Для виправлення помилки виділити цей фрагмент тексту й виконати команду **Сервис – Язык – Выбрать язык**. У вікні команди вибрати зі списку відповідна мова для перевірки правопису.

Для перевірки правопису у всьому документі або в його виділеному фрагменті виконати команду **Сервис – Правописание**.

# Лекція №4. ТЕКСТОВИЙ РЕДАКТОР MICROSOFT WORD. ГРАФІЧНІ ОБ'ЄКТИ. ТАБЛИЦІ

## 4.1. Команди пункту меню «Вставка»

Команда **Вставка – Разрыв** дозволяє розбити документ на окремі розділи, при цьому в кожному розділі може бути своя орієнтація й нумерація сторінок. Крім цього команда може бути використана для додавання нової сторінки (розділу).

Команда **Вставка – Номера страниц** дозволяє виконати автоматичну нумерацію сторінок документа. У вікні команди (рис. 4.1) вибирається положення номера (угорі або внизу сторінки) і спосіб його вирівнювання (ліворуч, праворуч, від центру та ін.). Якщо необхідно, щоб на першій сторінці документа номер не відображався, то вимикається режим «*Номер на першій сторінці*». Натиснувши кнопку **«Формат…»**, можна вказати, з якого числа повинна починатися нумерація сторінок у документі.

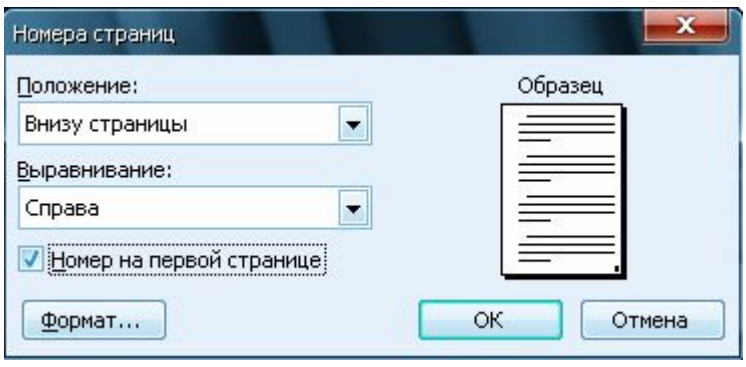



Рисунок 4.1. Вікно «Номера сторінок»

Номер сторінки розташовується у верхньому або нижньому колонтитулі. ***Колонтитул*** – це спеціальна область, яка розташована у верхньому або нижньому полі сторінки. Текст, який розташований у колонтитулі, буде автоматично повторюватися на всіх сторінках поточного розділу документа. Для активізації колонтитулів виконати команду **Вид – Колонтитул.** Для виходу з колонтитула натиснути кнопку **«Закрыть»** на панелі інструментів або 1 лкм у середині сторінки. Для видалення номерів сторінок необхідно відкрити відповідний колонтитул, виділити поле з номером сторінки й натиснути клавішу **Delete**.

Команда **Вставка – Дата и время** дозволяє додати в документ поле із системною датою й часом, які можуть автоматично обновлятися при відкритті документа.

Команда **Вставка – Символ** дає можливість додати в документ символи, які не можна ввести із клавіатури (грецькі літери, значки, спеціальні знаки та ін.). У вікні команди можна вибрати шрифт (найбільше часто використовуються символи які містяться в шрифті **Symbol**), потім знайти й виділити необхідний символ і натиснути кнопку **«Вставить»**.

Команда **Вставка – Ссылка – Сноска** для попередньо виділеного слова створює виноску. У вікні команди потрібно вибрати положення звичайної виноски (текст виноски буде поміщений унизу сторінки або внизу тексту) або кінцевої виноски (текст виноски буде поміщений наприкінці документа або розділу), а також задати формат номера. Потім у відповідній позиції ввести текст виноски.

Команда **Вставка – Ссылка – Оглавление и указатели** призначена для створення автоматичного змісту документа. Перед її використанням необхідно:

1) виконати нумерацію сторінок документа;

2) у документі кожному заголовку, що повинен бути поміщений у зміст, присвоїти стиль «Заголовок 1», «Заголовок 2» та ін., залежно від його рівня в ієрархії змісту;

3) установити КППВ у позицію, де буде починатися зміст документа, виконати команду, вибрати вкладку «Зміст», задати формат і інші необхідні параметри змісту.

Отриманий зміст можна форматувати як звичайний текст. Для переходу до необхідного пункту в документі натиснути клавішу **Ctrl** і, утримуючи її, виконати 1 лкм на необхідному пункті змісту. Для оновлення виділити зміст і з контекстного меню вибрати команду «**Обновить поле**». У вікні команди вибрати режим оновлення: *обновити тільки номера сторінок* або *обновити цілком* (якщо заголовки в тексті були змінені).

Команда **Вставка – Файл** дозволяє вставити в поточний документ повний вміст іншого текстового файлу.

Команда **Вставка – Объект** використовується для запуску допоміжних програм:

1) **Microsoft Equation** – редактор формул, що призначений для введення математичних формул будь-якого рівня складності.

2) **Microsoft Graph** – програма призначена для побудови діаграм.

3) **Organization Chart** – організаційна діаграма, що призначена для побудови структурних схем (наприклад, структура підприємства).

## 4.2. Робота із графічними об'єктами

### 4.2.1. Загальні положення

У редакторі MS Word є набір стандартних рисунків, які об'єднані в колекцію Microsoft Office. Всі рисунки, які утворять дану колекцію, розділені на категорії. Для додавання рисунку в документ потрібно виконати команду **Вставка – Рисунок – Картинки**. На додатковій панелі вибрати режим *Колекція Microsoft Office*. У вікні, що з'явилося, вибрати категорію колекції, у правій частині вікна виділити рисунок і перетягнути його за допомогою ЛКМ на сторінку документа. Для додавання рисунку, що зберігається в зовнішньому файлі, виконати команду **Вставка – Рисунок – Из файла**.

Рисунок у документі можна за допомогою ЛКМ переміщати, обертати й змінювати його розміри (рис. 4.2)

Щоб обрізати частину рисунку необхідно включити панель настроювання зображення (команда **Вид – Панели инструментов – Настройка изображения** або з контекстного меню рисунку вибрати команду «**Добавить панель настройки изображения**»). На панелі інструментів натиснути кнопку «**Обрезка**»  , установити курсор миші у вигляді  на один з маркерів границь малюнка , натиснути ЛКМ і втримуючи її обрізати частину рисунку.

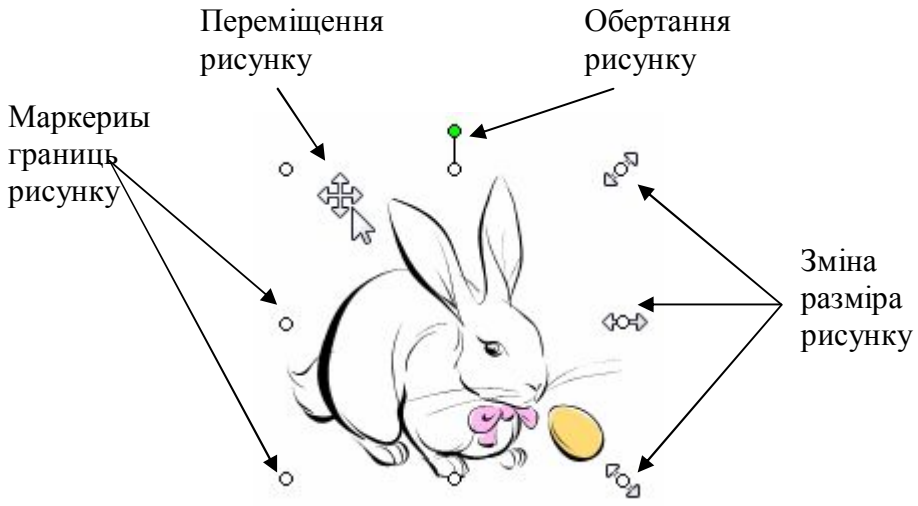



Рисунок 4.2. Дії над рисунком

Рисунок, як і будь-який інший графічний об'єкт редактора, має набір ***властивостей*** (параметрів) – розмір, масштаб відображення, яскравість і контрастність, положення на сторінці, обтікання та ін. Для перегляду й зміни властиво-

стей рисунку виділити його й виконати команду **Формат – Рисунка** (або з контекстного меню рисунку вибрати команду «**Формат рисунка**»). Кожному набору властивостей відповідає певна вкладка вікна команди.

Одним з основних властивостей рисунку (графічного об'єкта) є *обтікання* – спосіб розташування тексту щодо малюнка.

Найбільше часто використовуються наступні способи обтікання рисунку текстом (рис. 4.3):

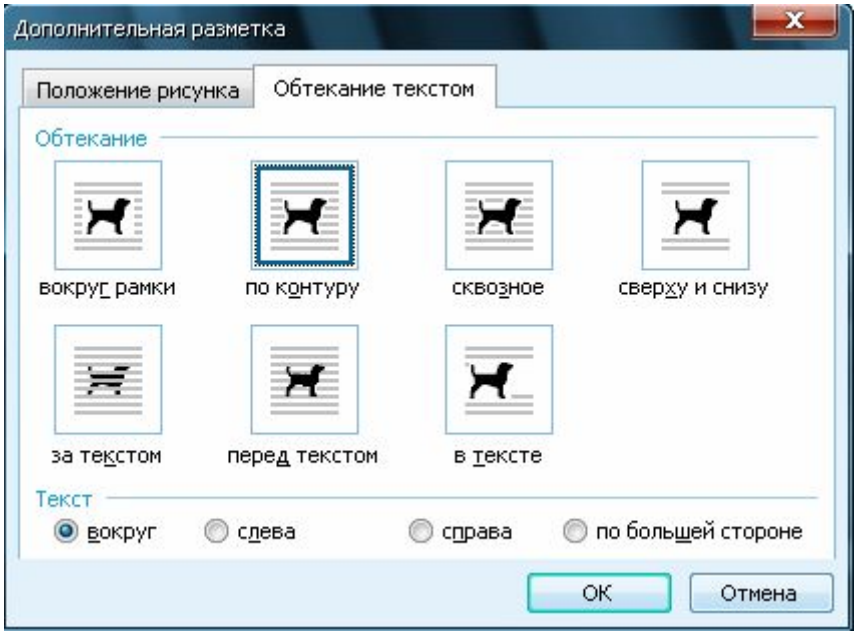



Рисунок 4.3. Способи обтікання рисунку текстом

*перед текстом* – цей спосіб дозволяє розташовувати рисунок у будь-якій позиції на сторінці в не залежності від розташування тексту;

*навколо рамки* або *по контуру* – текст може розташовуватися навколо рисунку певним способом, тобто переміщення рисунку по сторінці буде викликати зміну розташування тексту;

*зверху й знизу* – текст може розташовуватися тільки зверху й знизу щодо рисунку;

*у тексті* – рисунок може розташовуватися тільки в певних позиціях на сторінці й сприймається як елемент тексту.

Так само у вікні команди можна вибрати спосіб вирівнювання рисунку по горизонталі на сторінці. Для перегляду всіх можливих способів обтікання необхідно натиснути кнопку «**Дополнительно**», і в новому вікні вибрати вкладку **Обтікання текстом**. На цій вкладці можна задати додаткові обмеження на розташування тексту щодо рисунку.

### 4.2.2. Об'єкт Wordrt

Дозволяє додавати в документ фігурні написи. Для його створення необ-

хідно виконати команду **Вставка – Рисунок – Объект WordArt**. На першому кроці вибирається зовнішній вигляд напису, на другому кроці вводиться текст напису. До об'єкта WordArt застосовні ті ж дії, що й до звичайного малюнка.

### 4.2.3. Малювання за допомогою автофігур

***Автофігури*** – це набір найпростіших геометричних фігур (лінія, овал, прямокутник та ін.). Всі автофігури розділені на категорії: *лінії*, *основні фігури*, *фігурні стрілки*, *блок-схема* та ін. Для використання автофігур необхідно включити панель інструментів «*Малювання*» (команда **Вид – Панели инструментов – Рисование**) (рис. 4.4).


Рисунок 4.4. Панель інструментів «Малювання»

Щоб намалювати автофігуру необхідно на панелі інструментів вибрати (1 лкм) необхідну автофігуру (кнопка «**Автофигуры**»). Підвести курсор миші у вигляді «+» у необхідну позицію на сторінці, нажати ЛКМ і, утримуючи її, розтягти фігуру до необхідного розміру. Якщо при цьому натиснути клавішу **Shift**, то з овалу вийде коло, із прямокутника – квадрат, лінії й стрілки будуть строго вертикальними або горизонтальними.

Для виділеної автофігури можна за допомогою відповідної кнопки на панелі «*Малювання*» змінити тип і товщину лінії, цвіт лінії й заливання, додати тінь або об'єм. Також автофігура має звичайний набір властивостей, які мають графічні об'єкти (п. 4.2.1).

Усередині більшості автофігур можна розташовувати текст. Для цього необхідно з контекстного меню автофігури вибрати команду «**Добавить текст**». Так само можна скористатися автофігурою *Напис*, у якій уже міститься область для введення тексту. Текст, що міститься усередині автофігури, можна форматувати звичайним способом. Для зміни напрямку тексту в автофігурі виділити його й виконати команду **Формат – Направление текста**.

Щоб видалити будь-який графічний об'єкт необхідно виділити його й натиснути клавішу **Delete**. Для видалення автофігури з текстом – виділити її (1 лкм на лінії фігури, а не в середині) і натиснути клавішу **Delete**.

### 4.2.4. Виділення графічних об'єктів

Існують різні способи одночасного виділення декількох графічних об'єктів:

1) за допомогою ЛКМ виділити перший графічний об'єкт, натиснути клавішу **Shift** і, утримуючи її, по черзі виділити інші об'єкти;

2) на панелі інструментів «*Малювання*» натиснути кнопку «Вибір об'єктів», потім натиснути ЛКМ і охопити рамкою всі виділювані об'єкти (рис 4.5).

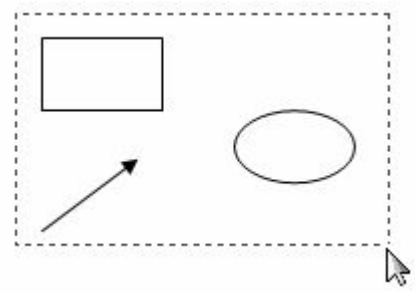


Рисунок 4.5. Виділення автофігур

Виділені автофігури можна згрупувати в єдиний об'єкт. Для цього необхідно виконати 1 пкм на виділеному й з контекстного меню вибрати команду «**Группировка – Группировать**» (рис. 4.6).

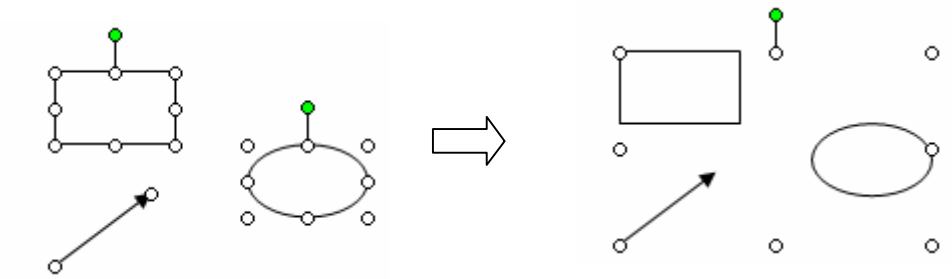


Рисунок 4.6. Автофігури до і після угрупування

Якщо графічні об'єкти перекривають один одного, то можна змінити порядок їхнього проходження. Для цього виконати 1 пкм на об'єкті й з контекстного меню вибрати команду «**Порядок – На передний план (На задний план** та ін.**)**».

## 4.3. Робота з таблицями

Створення таблиці виконується командою **Таблица – Вставить – Таблица**. У вікні команди необхідно вказати розмір таблиці – число стовпців і рядків. Способи виділення окремих елементів таблиці за допомогою миші показані на рис. 4.7. Також для виділення елементів таблиці можна використати команду **Таблица – Выделить – Таблица** (**Столбец**, **Строка** або **Ячейка**).

Для додавання рядків або стовпців у таблицю необхідно встановити КППВ у рядок або стовпець, біля яких будуть додаватися нові, і виконати команду **Таблица – Вставить – Строки выше** (**Строки ниже**, **Столбцы слева** або **Столбцы справа**). Для швидкого додавання рядка встановити КППВ за крайнім правим коміркою рядка й натиснути клавішу **Enter**.

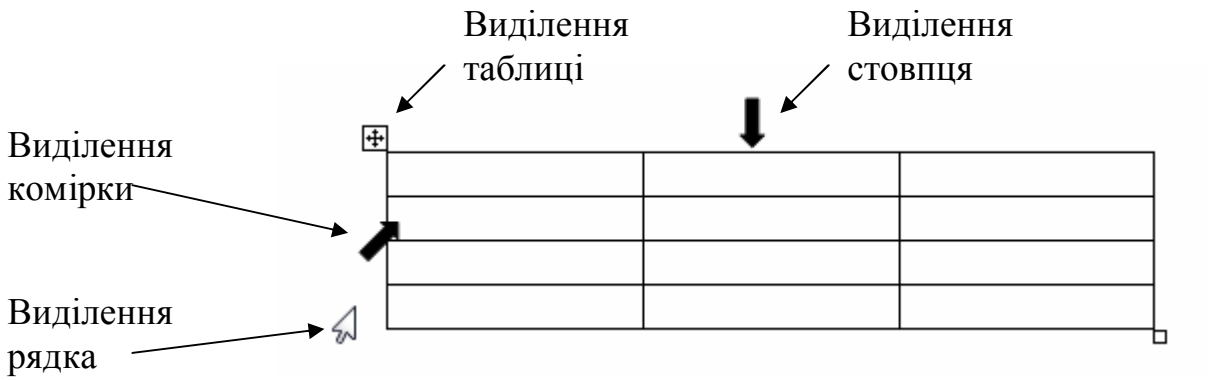



Рисунок 4.7. Виділення елементів таблиці

Для видалення рядків або стовпців з таблиці необхідно виділити їх і виконати команду **Таблица – Удалить – Строки** (**Столбцы**). Натискання клавіші **Delete** видаляє тільки вміст виділених комірок, а не елементи таблиці.

Для зміни висоти рядків або ширини стовпців таблиці досить установить курсор миші на відповідну границю (курсор повинен прийняти вид двунаправленої стрілки), натиснути ЛКМ і, утримуючи її, змінити необхідний розмір.

Для вирівнювання ширини декількох стовпців або висоти декількох рядків потрібно виділити їх і виконати команду **Таблица – Автоподбор – Выровнять высоту строк** (**Выровнять ширину столбцов**).

Для вирівнювання вмісту комірок по горизонталі й вертикалі необхідно їх виділити й з контекстного меню вибрати команду **«Выравнивание в ячейке»**.

При створенні таблиць зі складною структурою спочатку уставляється таблиця з максимально необхідною кількістю стовпців і рядків, а потім таблиця доводиться до заданого вигляду шляхом об'єднання комірок (команда **Таблица – Объединить ячейки**).

Для зміни зовнішнього вигляду границь таблиці, виділити необхідні комірки й виконати команду **Формат – Границы и заливка** (див. п. 3.5.5) або скористатися панеллю інструментів «*Таблиці й границі*».

## 4.4. Друк документа

Перед виконанням друку документа рекомендується виконати попередній його перегляд (команда **Файл – Предварительный просмотр)**, тобто подивитися, як буде розташовуватися вміст документа на папері.

Для друку документа потрібно виконати команду **Файл – Печать**. При цьому є можливість вибрати, яку частину документа необхідно роздрукувати: весь документ, окремі сторінки або попередньо виділений фрагмент.

# Лекція №5. ОСНОВИ РОБОТИ В MICROSOFT EXCEL

## 5.1. Основні теоретичні положення

Табличний редактор Microsoft Excel призначений для обробки таблиць, виконання обчислень і побудови діаграм. Робочий файл редактора називається ***Книга*** й може складатися з одного або декількох ***Листів*** (рис. 5.1). Кожен лист являє собою таблицю, стовпці якої позначаються латинськими літерами, рядки – цифрами. На перетинанні стовпця й рядка знаходиться ***Комірка***. Кожен комірка має свою унікальну ***Адресу*** (наприклад: A1, AB345 і тобто.). Адреса поточної комірки завжди показана у полі Ім'я.

Декілька рядом розташованих комірок називаються ***Діапазоном***, що може бути прямокутним, у вигляді рядка або стовпця. Адреса діапазону складається з адрес лівої верхньої і правої нижньої комірок, які записані через : (наприклад, A2:C4, A3:E3, B1:B10 і тобто). При виділенні декількох несуміжних діапазонів необхідно тримати натиснутою клавішу **Ctrl**.

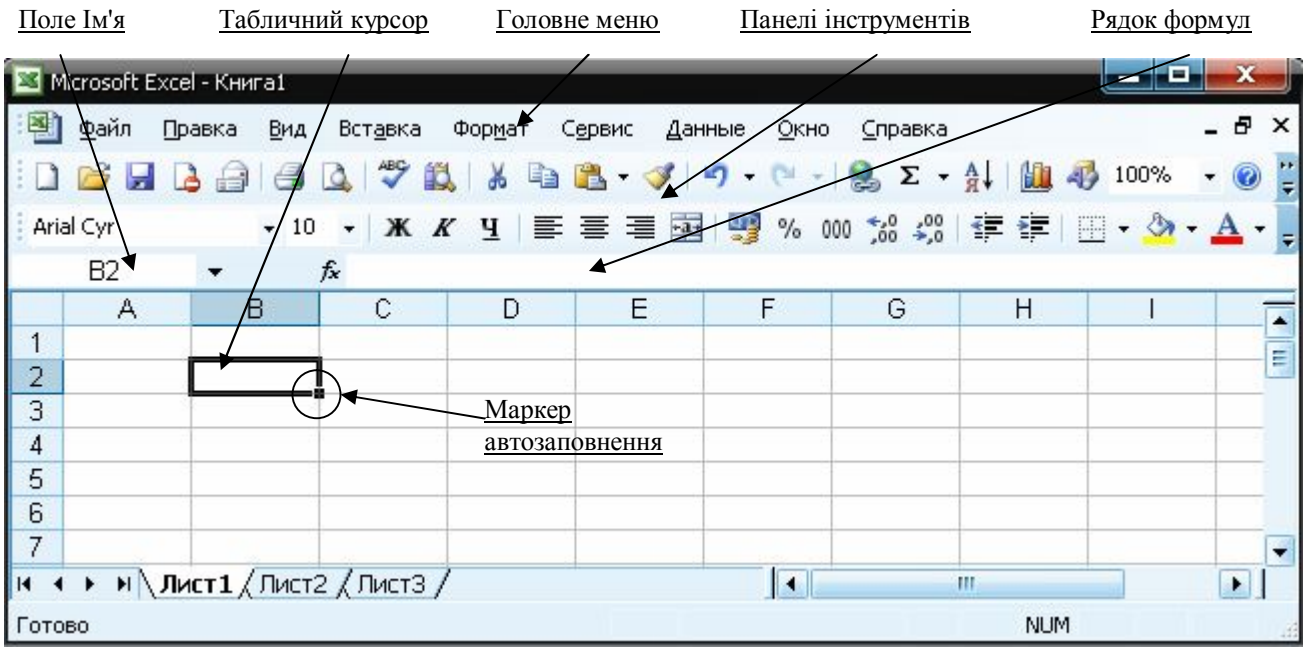



Рисунок 5.1. Вікно редактора Microsoft Excel.

## 5.2. Ввід і коректування даних, форматування комірок

Для вводу даних необхідно поставити табличний курсор у необхідну комірку, ввести значення із клавіатури і натиснути клавішу **Enter**. Дані бувають наступних типів: текст, числа, дати, формули й об'єкти.

Для заповнення діапазону комірок однаковими величинами необхідно: ввести в 1-у комірку необхідне значення, поставити курсор миші на ***Маркер автозаповнення*** (рис. 5.1), він прийме вигляд ┼, і розтягти область виділення на весь діапазон. Аналогічним способом заповнюються діапазони комірок, що

містять величини, яки перераховуються, тобто назви днів тижня й місяців, нумеровані списки (1-й клас, 2-й клас і тобто). Для заповнення комірок зростаючими послідовностями числових значень можна скористатися командою **Правка – Заполнить – Прогрессия**, перед викликом якої необхідно ввести 1-е значення послідовності й виділити заповнюваний діапазон.

Для коректування вмісту комірок можна або натиснути клавішу **F2** і внести зміни в самої комірці, або клацнути лівою кнопкою мишки в рядку формул і виконати там редагування даних.

При введенні тексту він може перекривати розташовані праворуч комірки, які насправді залишаються вільними. Тому при створенні таблиць заголовок кожного стовпця необхідно вводити не в першу «вільну» праворуч комірку, а в наступну після поточної, щоб у таблиці не залишалося порожніх стовпців. І тільки після введення всієї «шапки» таблиці рекомендується переходити до її форматування.

Під форматуванням розуміється зміна зовнішнього вигляду комірок і їхнього вмісту. Для цього необхідно виділити діапазон комірок (якщо форматується одна комірка, то в неї досить поставити табличний курсор), і виконати команду **Формат – Ячейки…** Вікно «Формат комірок» (рис. 5.2) містить наступні вкладки:

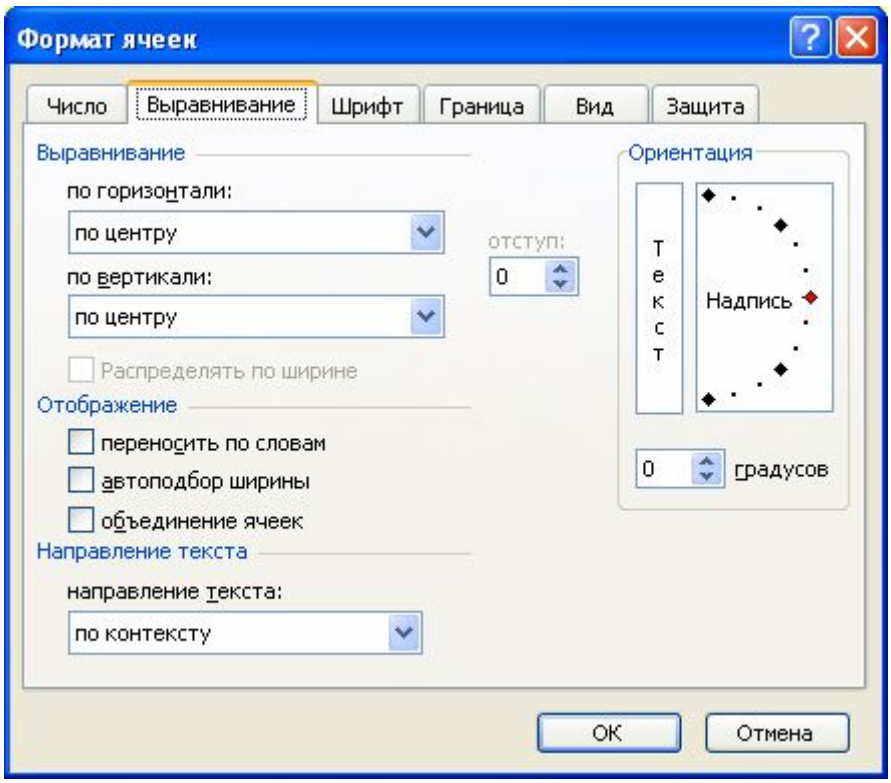



Рисунок 5.2. Вікно «Формат комірок»

**Число** – дозволяє встановлювати різні числові формати відображення даних:

*числовий формат* – найбільш загальний спосіб подання чисел, що дозво-

ляє змінювати кількість десяткових знаків у дробовій частині числа (розмірність числа);

*процентний формат* – множить значення комірки на 100 і виводить на екран зі знаком %;

*формат дата* – служить для відображення дат і часу, яки представлени числами, у вигляді дат.

**Вирівнювання –** дозволяє вирівнювати вміст комірок по вертикалі й горизонталі, змінювати орієнтацію тексту, поєднувати комірки й розташовувати текст комірки в кілька рядків (режим переносити за словами).

**Шрифт** – дозволяє змінювати параметри шрифту (тип, розмір, накреслення, ефекти).

**Границя** – дозволяє додавати коміркам границі.

**Вид** – дозволяє виконувати заливання комірок різними кольорами.

**Захист** – дозволяє захищати комірки листа від редагування або приховувати формули.

### 5.3. Використання вбудованих функцій для обчислень

Обчислення в MS Excel виконуються за допомогою ***формул***, які починаються зі знака =. Формула записується в один рядок, тому необхідно враховувати пріоритет виконання арифметичних операцій. Як аргументи формул можуть використовуватися числа, адреси комірок і функції. При створенні формули адреса комірки не вводиться із клавіатури, досить лише клацнути лівою кнопкою миші на необхідної комірки й посилання на неї з'явиться у виразі.

Наприклад, **=A5*C5/(B5^2)**, (вміст комірки А5 помножити на вміст комірки С5 і розділити на вміст комірки В5, яке піднесене в ступінь 2).

Для обчислення суми значень діапазону комірок зручно виконати наступні дії: поставити табличний курсор в комірку, у якої необхідно отримати результат; натиснути кнопку **Автосума** (**Σ**) на панелі інструментів; виділити необхідні комірки й натиснути клавишу **Enter**.

Для обчислень за допомогою стандартних функцій MS Excel необхідно: поставити табличний курсор в комірку, у якої потрібно отримати результат; виконати команду **Вставка – Функция**, або натиснути на панелі інструментів кнопку ***fx*** для виклику ***Майстра функцій***.

**Майстер** – це послідовність діалогових вікон, у кожному з яких необхідно вибрати або ввести необхідні параметри, для одержання кінцевого результату.

На 1-му кроці Майстра функцій необхідно вибрати категорію функції (для зручності використання, функції розділені на категорії), знайти й виділити функцію, потім натиснути **Ок**. Якщо невідомо категорію, до якої відноситься

функція, то спочатку рекомендується переглянути категорію **«10, що недавно використовувалися»**, а потім **«Повний алфавітний перелік»**.

На 2-му кроці Майстра функцій необхідно виділити адреси комірок або діапазонів, які будуть аргументами функції й натиснути **Ок**. Якщо це вікно закриває дані на аркуші, те його можна відсунути убік мишею.

При виконанні обчислень аналогічні формули не створюють заново, а отримують за допомогою автозаповнення.

Функції, які найбільш часто використовуються, приведені в табл. 5.1.

Таблиця 5.1

Функції MS Excel, які найбільш часто використовуються.

| № п/п | Функція | Категорія | Призначення |
|---|---|---|---|
| 1 | СУММ(число1; …) | Математичні | Обчислює суму значень діапазону комірок. |
| 2 | СРЗНАЧ(число1; …) | Статистичні | Обчислює середнє значення діапазону комірок. |
| 3 | МАКС(число1; …) | Статистичні | Знаходить максимальне значення в діапазоні комірок. |
| 4 | МИН(число1; …) | Статистичні | Знаходить мінімальне значення в діапазоні комірок. |
| 5 | СЧЕТ(знач1; …) | Статистичні | Підраховує кількість чисел у діапазоні комірок. |
| 6 | СЧЕТЕСЛИ(знач1; …) | Статистичні | Підраховує кількість чисел у діапазоні комірок, які задовольняють заданої умові. |
| 7 | СУММПРОИЗВ(мас1; …) | Математичні | Перемножує відповідні елементи заданих масивів (діапазонів) і повертає суму добутків. |

В MS Excel існують три види адресації комірок – відносна, абсолютна й змішана.

У випадку ***відносної адресації*** адреса комірки задається звичайним способом (наприклад, A3, C12 і тобто). При копіюванні формул за допомогою маркера автозаповнення, відносні адреси яки містяться в них, будуть змінюватися. Якщо формула «розтягується» по рядку, то в адресі буде змінюватися номер стовпця. Якщо по стовпцю, то номер рядка.

У випадку ***абсолютної адресації*** у адресі комірки перед літерою й цифрою додається знак $ (наприклад, $A$3, $C$12 і тобто). При копіюванні формул за допомогою маркера автозаповнення, абсолютні адреси, яки містяться в них, змінюватися не будуть. Виходить, якщо в аналогічних формулах повинна стоя-

ти той сама адреса, то її треба зробити абсолютною. Для цього після вводу адреси у формулу необхідно натиснути **F4** (на адресу додадуться знаки $).

У випадку ***змішаної адресації*** знак $ додається перед літерою або перед цифрою в адресі комірки (наприклад, $A3, C$12 і тобто). При копіюванні формули за допомогою маркера автозаповнення не буде змінюватися номер стовпця або рядка в змішаній адресі, перед яким стоїть знак $.

**Приклад 5.1**. Обчислити функцію Y = Cos (π/2·X)/A, при X∈[0;1], hX=0,2. Для цього в діапазон A2:A12 за допомогою **Правка – Заполнить – Прогрессия** занесемо значення X із заданого інтервалу.

| | A | B | C | D |
|---|---|---|---|---|
| **1** | **X** | **Y (значення)** | **Y (формули)** | **A** |
| **2** | 0 | 0,8333 | =COS(ПИ()/2*A2)/$D$2 | 1,2 |
| **3** | 0,2 | 0,7925 | =COS(ПИ()/2*A3)/$D$2 | |
| **4** | 0,4 | 0,6742 | =COS(ПИ()/2*A4)/$D$2 | |
| **5** | 0,6 | 0,4898 | =COS(ПИ()/2*A5)/$D$2 | |
| **6** | 0,8 | 0,2575 | =COS(ПИ()/2*A6)/$D$2 | |
| **7** | 1 | 0 | =COS(ПИ()/2*A7)/$D$2 | |

В комірку B2 введемо формулу для обчислення Y (формула приведена в комірці C2). Для обчислення наступних значень Y у стовпці B будуть використовуватися формули аналогічні, уведеної в комірку B2. При цьому адреса комірки, що містить значення X, повинна змінюватися в кожній наступній формулі (використовується відносна адреса), а адреса комірки, яка містить значення A, повинна залишатися незмінною в кожній наступній формулі (використовується абсолютна адреса).

У ході обчислень часто доводиться вибирати одне з декількох значень залежно від умов. Для цих цілей можна використовувати функції з категорії **«Логічні»**.

Функція **И** (логічне І) має наступний синтаксис:

**=И (логічне_значення1; логічне_значення2; ...)**

**Логічне_значення1, логічне_значення2, ...** — це від 1 до 30 умов, що перевіряють, які можуть мати значення або ІСТИНА, або НЕПРАВДА.

Функція **ИЛИ** (логічне АБО) має наступний синтаксис:

**= ИЛИ(логічне_значення1; логічне_значення2; ...)**

Функція **НЕ (логічне** заперечення). Синтаксис написання:

= **НЕ(**логічне_значення1; логічне_значення2;…)...

Змінює на протилежне логічне значення свого аргументу. Функція **НЕ** використовується в тих випадках, коли необхідно бути впевненим у тому, що значення не дорівнює деякій конкретній величині.

Функція **ЕСЛИ** має наступний синтаксис:

**=ЕСЛИ(лог_вираз; значення_якщо_істина; значення_якщо_неправда)**

**Лог_вираз** — це будь-яке значення або вираз, що приймає значення ІСТИНА або НЕПРАВДА.

**Значення_якщо_істина** — це значення, що повертається, якщо лог_вираз дорівнює ІСТИНА. Якщо лог_вираз дорівнює ІСТИНА, а значення_якщо_істина порожньо, то повертається значення 0. Значення_якщо_істина може бути формулою.

**Значення_якщо_неправда** — це значення, що повертається, якщо лог_вираз дорівнює НЕПРАВДА. Якщо лог_вираз дорівнює НЕПРАВДА, а значення_якщо_неправда опущена (тобто після значення_якщо_істина немає крапки з комою), те повертається логічне значення НЕПРАВДА. Якщо лог_вираз дорівнює НЕПРАВДА, а значення_якщо_неправда порожньо (тобто після значення_якщо_істина стоїть крапка з комою з наступною закриваючою скобкою), те повертається значення 0. Значення_якщо_неправда може бути формулою.

Функції ЕСЛИ можуть бути вкладені друг у друга як значення аргументів значення_якщо_істина й значення_якщо_неправда для конструювання більш складних перевірок.

## 5.4. Побудова діаграм

Діаграми будуються на основі числових даних, які називаються ***рядами***. Кожний ряд має ім'я (за замовчуванням це Ряд1, Ряд2 та ін.) і являє собою рядок або стовпець із даними. Перед побудовою діаграми бажано виділити вихідні дані із заголовками для рядків або стовпців, тобто визначити імена для рядів. Потім виконати команду **Вставка – Діаграма** або натиснути кнопку **Майстер діаграм**.

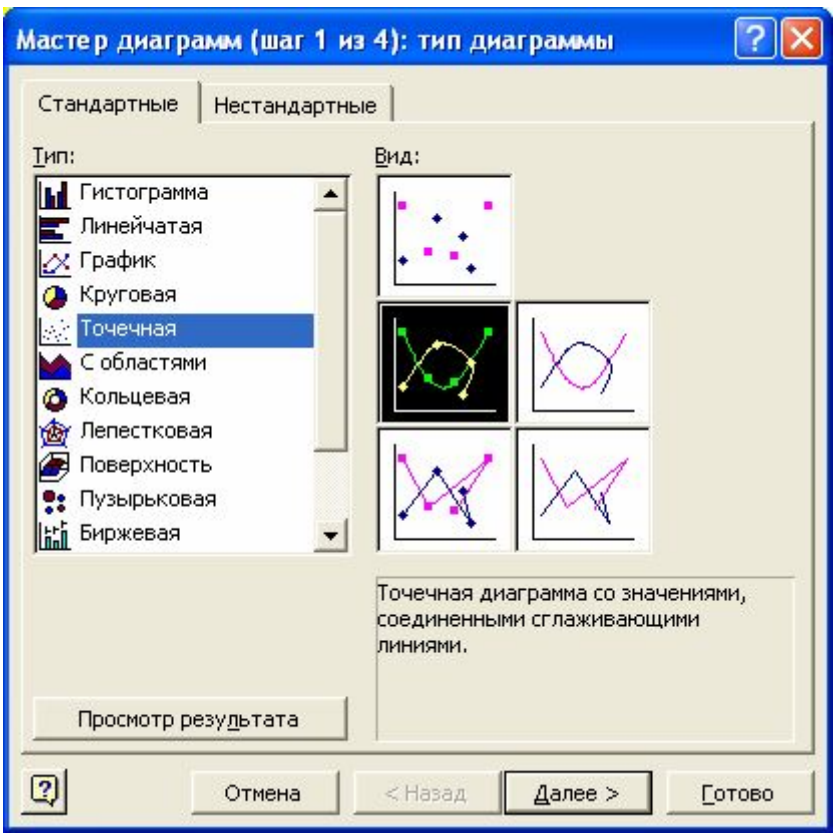



Рисунок 5.3. Вікно Майстра діаграм (крок 1).

На 1-му кроку майстра діаграм вибирається **Тип діаграми** і її **Вид** (рис. 5.3). Наприклад, для

побудови графіка функції **Y=Sin(X)** необхідно вибрати тип діаграми **Точкова**, а не **Графік**. При натисканні кнопки **Перегляд результату** можна побачити майбутню діаграму.

На 2-му кроку майстра діаграм на вкладці «**Діапазон даних»** вибирається розташування рядів даних (у рядках або в стовпцях) таким чином, щоб у ***Легенді*** виводилися назви рядів даних, а не стандартні імена (рис. 5.4). На вкладці «**Ряд**» є можливість за допомогою відповідних кнопок додати новий або видалити наявний ряд даних (при додаванні ряду необхідно на листі виділити комірку, що містить назву ряду, і діапазон, що містить значення). Для багатьох типів діаграм можна вказати діапазон комірок, що містить значення для підпису осі X.

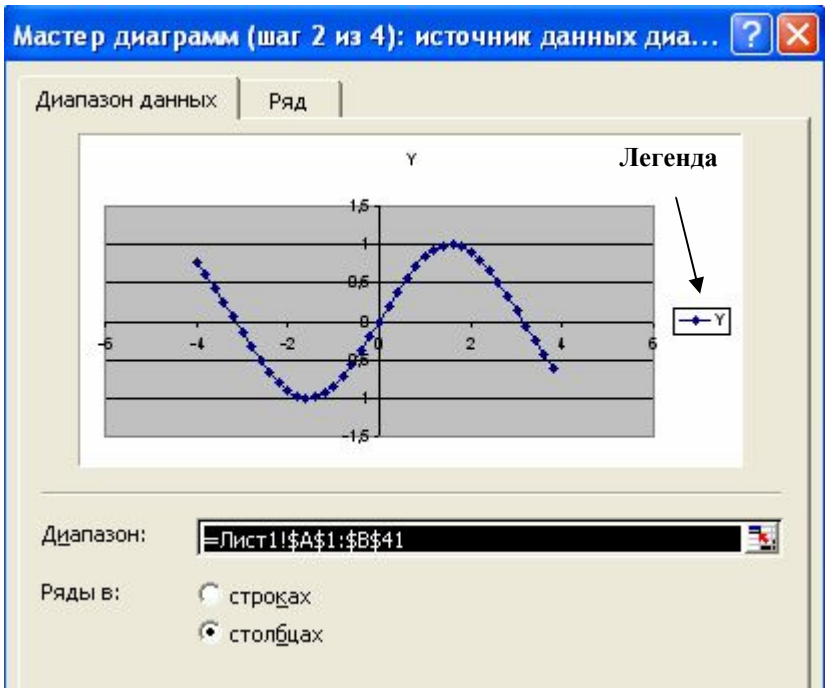



Рисунок 5.4. Вікно Майстра діаграм (крок 2).

На 3-му кроку майстра діаграм уводяться назви діаграми й осей, а також підписи даних і інші параметри діаграми.

На 4-му кроку майстра діаграм вибирається розміщення діаграми на одному з існуючих листів, або на окремому новому листі. У другому випадку діаграма буде виведена на весь лист.

У створеній діаграмі можна форматувати (змінювати зовнішній вигляд і параметри) будь-який її елемент (область побудови, легенда, осі та ін). Для цього необхідно клацнути правою клавішею миші на елементі й з контекстного меню вибрати команду **Формат…** (назва елемента). Також є можливість змінити обрані на кожному кроці майстра діаграм параметри, виконавши команду **Діаграма –** (Назва кроку майстри діаграм).

## 5.5. Макрос

Макрос - це один раз виконана й збережена послідовність будь яких дій, (форматування зовнішнього вигляду комірок, ввід даних, обчислення, копіювання й тобто).

Макрос являє собою програму на Visual Basic. Створений макрос можна багаторазово виконувати. Щоб макрос працював не з однією поточною коміркою, а з будь-яким діапазоном комірок необхідно перед створенням макросу виділити діапазон комірок, виконати команду: **Сервис – Макрос – Начать запись**, потім у вікні, що з'явилося, ввести ім'я макросу, задати комбінацію клавіш для швидкого виклику макросу й натиснути кнопку Ok. Всі подальші дії, що виконуються, будуть записуватися в макрос. Після виконання необхідних операцій виконати команду: **Сервис – Макрос – Остановить запись**.

Для застосування макросу виділити діапазон комірок для якого треба виконати макрос, виконати команду: **Сервис – Макрос – Макросы.**

У вікні, що з'явилося (рис. 9.5), виділити ім'я необхідного макросу, натиснути кнопку **Виконати**. При відкритті файлу, що містить макроси, буде з'являтися вікно - попередження про те, що у файлі є макроси, натиснути кнопку не відключати макроси.

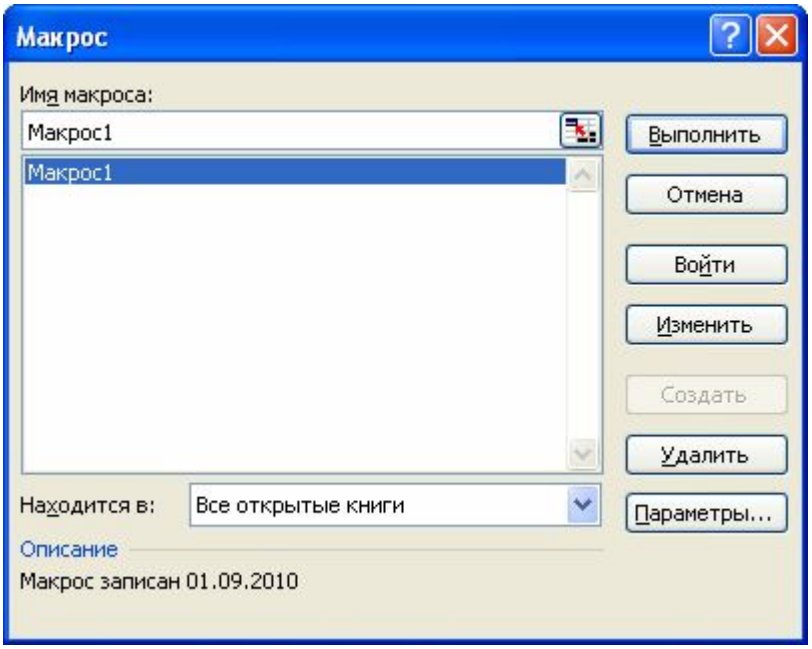



Рисунок 9.5. Запуск макросу на виконання

## 5.6. Приклад виконання лабораторної роботи «Використання убудованих функцій Excel. Організація вкладених циклів в Excel»

**Завдання** 1. Обчислити значення функції, за допомогою убудованих функцій Excel **ЕСЛИ** й **И.** Побудувати графік функції $y = f(x)$.

$$y=\begin{cases}2x, & якщо \quad -1\le x<0\\ x^2, & якщо \quad 0\le x\le 1\\ -x+1, & якщо \quad 1<x\le 2\end{cases} \qquad при \quad \Delta x=0{,}1$$

**Порядок дій.** Ввести в діапазон A2:A32 за допомогою автозаповнення значення змінної **x** з діапазону [-1,2] із кроком Δx=0,1. Потім в комірку B2 ввести формулу:

=ЕСЛИ(И(A2≥-1; A2<0); 2*A2; ЕСЛИ(И(A2≥0; A2≤1); A2^2; -A2+1)),

і скопіювати її за допомогою автозаповнення в діапазон B3:B32.

| | A | B | C |
|---|---|---|---|
| **1** | **x** | **y** | y (формули) |
| **2** | -1 | -2 | =ЕСЛИ(И(A2>=-1;A2<0);2*A2;ЕСЛИ(И(A2>=0;A2<=1);A2^2;-A2+1)) |
| **3** | -0,9 | -1,8 | =ЕСЛИ(И(A3>=-1;A3<0);2*A3;ЕСЛИ(И(A3>=0;A3<=1);A3^2;-A3+1)) |
| **4** | -0,8 | -1,6 | =ЕСЛИ(И(A4>=-1;A4<0);2*A4;ЕСЛИ(И(A4>=0;A4<=1);A4^2;-A4+1)) |
| **5** | -0,7 | -1,4 | =ЕСЛИ(И(A5>=-1;A5<0);2*A5;ЕСЛИ(И(A5>=0;A5<=1);A5^2;-A5+1)) |
| **6** | -0,6 | -1,2 | =ЕСЛИ(И(A6>=-1;A6<0);2*A6;ЕСЛИ(И(A6>=0;A6<=1);A6^2;-A6+1)) |
| **…** | … | … | … |
| **32** | 2 | -1 | =ЕСЛИ(И(A32>=-1;A32<0);2*A32;ЕСЛИ(И(A32>=0;A32<=1);A32^2;-A32+1)) |

**Завдання** 2. Обчислити значення функції: $Z=x^2 y - x$, при $-2 \le x \le 4$, $\Delta x=1$ і $-2 \le y \le 2$, $\Delta y=1$.

**Порядок дій.** Ввести в діапазон B1:H1 за допомогою автозаполнения значення змінної **x** з діапазону [-2,4] із кроком Δx=1, і в діапазон A2:A6 значення змінної **y** з діапазону [-2,2] із кроком Δy=1. Потім в комірку B2 ввести формулу:

=B$1^2*$A2 - B$1,

використовуя змішану адресацію. Скопіювати формулу за допомогою автозаполнения в діапазон B2:H6.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| **1** | **y \ x** | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
| **2** | -2 | -6 | -1 | 0 | -3 | -10 | -21 | -36 |
| **3** | -1 | -2 | 0 | 0 | -2 | -6 | -12 | -20 |
| **4** | 0 | 2 | 1 | 0 | -1 | -2 | -3 | -4 |
| **5** | 1 | 6 | 2 | 0 | 0 | 2 | 6 | 12 |
| **6** | 2 | 10 | 3 | 0 | 1 | 6 | 15 | 28 |

# Лекція №6. ПРОГРАМНІ ЗАСОБИ РОБОТИ З БАЗАМИ ДАНИХ

## 6.1. Проектирование базы данных

### 6.1.1. Понятие базы данных

Под базой данных понимают совокупность взаимосвязанных данных, которую обрабатывают с помощью СУБД (системы управления базами данных). Простейшим примером базы данных может служить записная книжка, в которую заносят фамилии, адреса, телефоны… База данных создается, редактируется и обрабатывается с помощью СУБД.

База данных в Access – единый большой объект, который объединяет такие основные составляющие как таблицы, отчеты, запросы, формы и позволяет их хранить в едином дисковом формате.

В Microsoft Access прежде чем создавать таблицы, формы и другие объекты необходимо задать структуру базы данных. Хорошая структура базы данных является основой для создания адекватной требованиям, эффективной базы данных.

Основным структурным компонентом БД является таблица, содержащая записи определенного вида и формы. Каждая запись таблицы содержит всю необходимую информацию об отдельном элементе БД. Столбцы в базе данных называются полями.

Практически все базы данных поддерживают реляционную модель – набор связанных таблиц. Каждая таблица должна при этом иметь уникальное имя, порядок следования строк и столбцов внутри таблиц безразличен.

### 6.1.2. Проектирование базы данных

Ниже приведены основные этапы проектирования базы данных:

1. Определение цели создания базы данных.
2. Определение таблиц, которые должна содержать база данных.
3. Определение необходимых в таблице полей.
4. Задание индивидуального значения каждому полю.
5. Определение связей между таблицами.
6. Обновление структуры базы данных.
7. Добавление данных и создание других объектов базы данных.
8. Использование средств анализа в Microsoft Access.

Прежде, чем создавать базу данных, необходимо спроектировать ее на бумаге. Первым этапом при создании таблицы является определение перечня полей, из которых она должна состоять, их типов и размеров (для текстовых полей). При этом каждому полю присваивается уникальное имя. Указываемый тип данных показывает СУБД, каким образом нужно обрабатывать поле.

*Некоторые советы при проектировании базы данных:*

- Если информация в некоторых полях повторяется в ряде записей, следует подумать и помещении ее в отдельную таблицу вместе с тем полем, по которому будет осуществляться связь;
- Следует сразу же определить ряд полей, в которых не допускаются повторения, для задания им первичного ключа или индекса.
- Не рекомендуется включать в таблицу данные, которые являются результатом вычислений. Вычисляемые поля будут входить в запросы, формы и отчеты.
- Информацию следует разбивать на наименьшие логические единицы (Например, поля «Имя» и «Фамилия», а не общее поле «ФИО»).
- Поля таблиц, по которым будет осуществлять связь, должны быть обязательно одного типа.

### 6.1.3. Пример проектирования базы данных

В качестве примера разберем базу данных, ведущую учет розничной продажи журналов в нескольких пунктах продажи. Сведения, которые будут представлены в базе данных: дата продажи, название журнала, его цена, издательство, количество проданных журналов, адрес пункта продажи, ФИО ответственного лица.

В этой базе имеются повторения: сведения о журнале – их выносим в отдельную таблицу ***Журналы***; сведения о пункте продажи – будет таблица ***Пункт продажи***, и сведения об издательстве – таблица ***Издательство***.

Для связи между таблицами введем числовые поля ***КодЖурнала, КодИздательства*** и ***КодПункта***. Анализируя типы полей, примем такие таблицы:

Таблица 6.1.1. **Журналы**

| ***Название поля*** | ***Тип данных*** |
|---|---|
| КодЖурнала | Числовой |
| Название | Текстовый |
| КодИздательства | Числовой |
| Цена | Денежный |

Таблица 6.1.2 **Издательство**

| ***Название поля*** | ***Тип данных*** |
|---|---|
| КодИздательства | Числовой |
| Название | Текстовый |
| Адрес | Текстовый |
| Телефон | Текстовый |

Таблица 6.1.3. **Пункт продажи**

| ***Название поля*** | Тип данных |
|---|---|
| КодПункта | Числовой |
| Адрес | Текстовый |
| Фамилия | Текстовый |
| Имя | Текстовый |

Таблица 6.1.4. **Продажи**

| ***Название поля*** | Тип данных |
|---|---|
| Дата продажи | Дата/Время |
| КодЖурнала | Числовой |
| КодПункта | Числовой |
| Количество | Числовой |

**Определяем поля, в которых не допускаются повторения: в таблице** *Журналы* **это поле** *КодЖурнала***, в таблице** *Издательства* **– поле** *КодИзда-*

*тельства*, **в таблице** *Пункт Продажи* – **поле** *КодПункта*. **В дальнейшем этим полям необходимо задать первичный ключ (сделать их ключевыми).**

### 6.1.4. Знакомство с окном базы данных

Приступим к созданию базы данных. Все объекты, относящиеся к базе данных, Access хранит в одном большом файле. Именно этот файл следует открывать для дальнейшей работы.

Запустите программу **Microsoft Access** (**Пуск** – **Программы**). При запуске открывается диалоговое окно, в котором предлагается создать новую базу данных или открыть существующую. Если появляется это диалоговое окно, выберите параметр **Новая база данных,** а затем нажмите кнопку **ОК**. В следующем окне следует указать имя создаваемой базы данных и ее место сохранения. После нажатия кнопки **Сохранить** откроется окно, изображенное на рис.6.1.1.

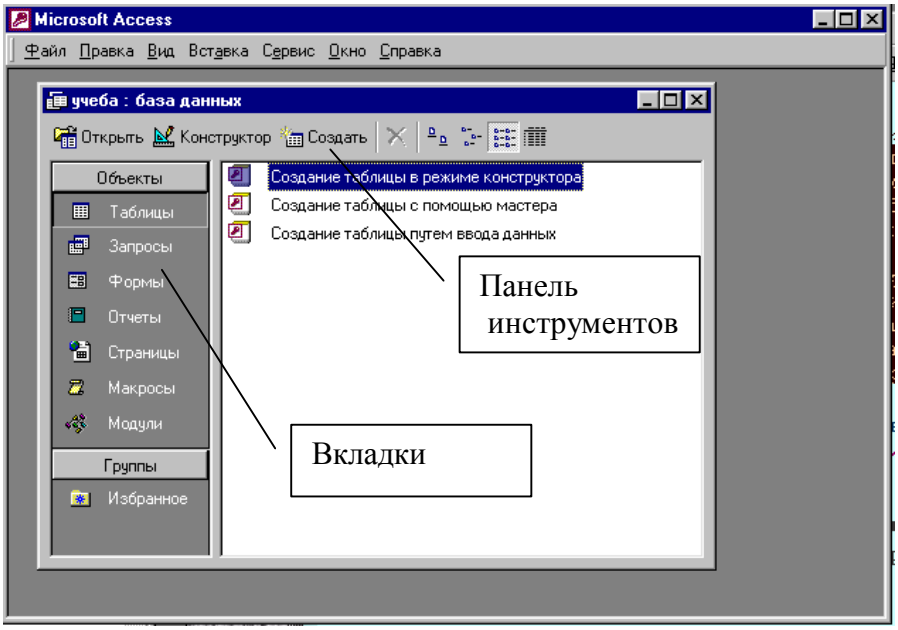



Рисунок. 6.1.1. Окно программы Microsoft Access

Если база данных уже была открыта или уже закрыто окно запуска, нажмите кнопку **Создать базу данных** на панели инструментов. В открывшемся окне дважды нажмите кнопку мыши, установив указатель на значок новой базы данных. В диалоговом окне создания файла из раскрывающегося списка **Папка** выберите нужный диск, а затем зайдите в личную папку. В поле **имя файла** введите имя базы данных, в нашем примере база данных будет называться **Обучение** (расширение автоматически присваивается **mdb**), нажмите кнопку

**Создать**.

Окно базы данных состоит из семи вкладок (объекты ACCESS) (рис.6.1.1):

- ✓ **Таблицы -** используется для задания структуры и хранения данных.

- ✓ **Запросы -** с его помощью проводят поиск информации, отвечающей заданным условиям.
- ✓ Формы - **объект, предназначенный для ввода данных, расчетов и проявления талантов дизайнера.**
- ✓ **Отчеты -** создание выходной формы для просмотра и распечатки.
- ✓ **Макрос -** структурированное описание последовательности действий.
- ✓ **Модуль -** программа на языке BASIC.

Создание базы данных начнем с создания таблиц, поэтому выберите вкладку Т**аблицы.**

## 6.2. Создание таблиц

Таблицы можно создавать несколькими способами.

В комплект поставки Microsoft Access входит мастер таблиц, который облегчает создание многих типовых таблиц на основе шаблона. Мастер таблиц последовательно пройдет с вами процесс создания новой таблицы, вы просто находите таблицу, ближе всего соответствующую типу данных, с которыми вы хотите работать, и затем выбираете нужные поля.

Таблицы также можно создавать в режимах ***Конструктора*** или табличном режиме. В режиме конструктора происходит вся работа по определению и настройке структуры таблицы, как во время создания, так и при всех последующих модификациях. Режим таблицы рекомендуется использовать для создания простой таблицы или заполнения таблицы данными.

**!** ***Рекомендуется вначале создать таблицу в режиме конструктора, определив ее структуру, а затем, переключившись в режим таблицы, заполнять ее.***

Подробно структуру таблиц можно распечатать следующим образом: выберите **Сервис – Анализ Архивариус**. Выберите объект и укажите параметры, которые вы хотите включить в отчёт, выбрав их с использованием кнопки **Параметры**. Нажми **ОК**. Вы сможете предварительно посмотреть отчёт, а потом отпечатать.

### 6.2.1. Создание таблиц с помощью мастера.

Для выбора дважды щелкните по команде **Создание таблицы с помощью Мастера**. Появится окно Мастера таблиц.

Слева в диалогом окне находятся два переключателя: деловые или личные – установите нужный. В расположенном слева списке следует выбрать подходящую таблицу. В списке **Образцы полей** появятся все поля выбранной таблицы. Для добавления поля в создаваемую таблицу следует выделить его и нажать кнопку с одной стрелкой вправо. Для переноса всех полей нажмите

кнопку с двойной стрелкой вправо. Выбранные поля появятся в правой части диалогового окна. Если вы передумали добавлять поле, то выделите его в правой части и нажмите кнопку со стрелкой влево. Здесь же выбранное поле можно переименовать.

Нажмите кнопку **Далее** и задайте имя своей таблицы и выберите способ определения первичного ключа. В большинстве случаев эту задачу можно возложить на Microsoft Access.

На третьем шаге предлагается выбор из трех режимов:

1) *Изменить структуру таблицы* - таблица откроется в режиме **Конструктора**. Здесь можно поменять названия и тип полей.

2) *Ввести данные непосредственно в таблицу* – откроется окно режима **Таблицы** для просмотра и редактирования

3) *Ввести данные в таблицу с помощью формы, создаваемой мастером* – создается форма на основе создаваемой таблицы и открывается режим **Формы**. Вид формы самый простой – все поля располагаются друг под другом.

### 6.2.2. Создание таблиц в режиме конструктора.

Выберите режим **Создание таблицы в режиме конструктора** или щелкните по кнопке **Конструктор.** Каждая таблица состоит из **записей** (так называют строки) и **полей** (это столбцы).

Окно **Конструктора** (рис. 6.2.1) состоит из двух частей. В верхней части экрана задают структуру таблицы - названия полей, их тип и комментарии, если в них есть необходимость, нижняя часть экрана отводится под свойства полей. Причем, в какой строке стоит курсор, свойства такого поля и будут представлены. Количество строк в области **Свойства** зависит от выбранного типа поля вверху. Тип поля не вводится, а выбирается из раскрывающегося списка.

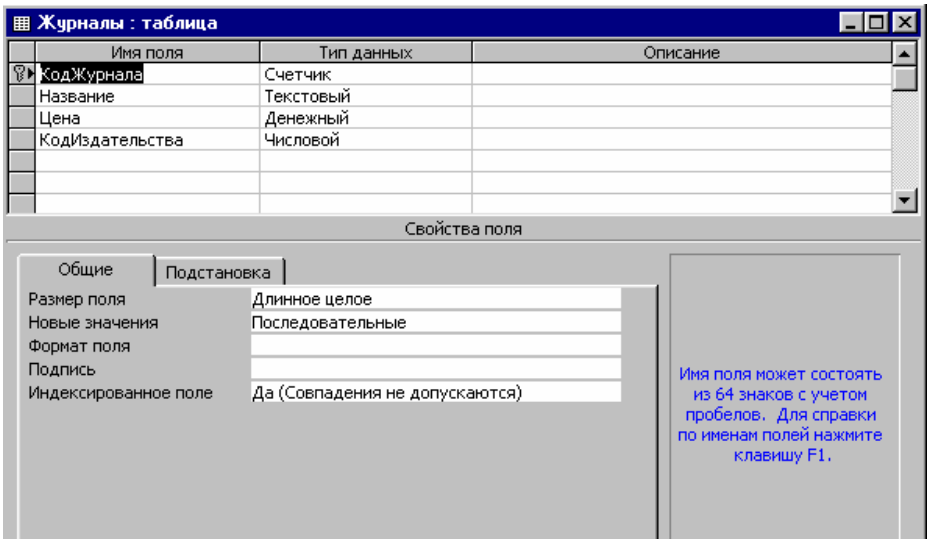



Рис. 6.2.1. Таблица *Журналы* в режиме конструктора

**Выбор типа данных.**

Допустимы поля следующих типов:

- ✓ ***Текстовый -*** предназначен для текстовой информации и чисел при невыполнении математических операций с ними с максимальной длиной ***до 255*** символов;
- ✓ ***Числовой*** - хранение числовых данных, над которыми будут выполняться математические операции, при работе с числовым полем можно изменить его внутренний формат с помощью свойства Размер поля;
- ✓ ***Поле Мето*** - предназначен для хранения произвольного текста, комментариев (до 64000 символов);
- ✓ ***Дата/Время*** - хранение данных указанных типов, при этом осуществляется контроль правильности ввода данных;
- ✓ ***Денежный*** - для проведения операции с денежными величинами;
- ✓ ***Логический(Да/Нет)*** - это величины, которые принимают **только два значения** - да/нет, Истина/Ложь и т.д., значение может быть представлено в виде флажка;
- ✓ ***Счетчик*** - специальное числовое поле, предназначенное для автоматического добавления уникального номера текущей записи в таблице данных;
- ✓ ***Поле объекта OLE -*** предназначен для хранения объекта, созданного другими приложениями (рисунок, звук, видеоклип, график, диаграмма).
- ✓ ***Мастер подстановок*** – заставляет поле принимать только те значения, которые перечислены в заранее определенном списке или в другой таблице.

Когда определяется имя и тип поля, в нижней области также можно задать ***свойства поля:***

***Размер поля*** - для текстовых полей определяет максимальную длину поля, а для числовых типов данных :

- ✓ *Байт*- целые числа от 0 до 255;
- ✓ *Целое* - целые числа от -32768 до 32767;
- ✓ *Длинное целое* -от $-2\cdot10^9$ до $2\cdot10^9$;
- ✓ *С плавающей точкой (4 байта)* - для чисел с дробной частью со значением поля от $-3{,}4\cdot10^{38}$ до $3{,}4\cdot10^{38}$;
- ✓ *С плавающей точкой (8 байт)* - для чисел с дробной частью со значением поля от $-1{,}8\cdot10^{308}$ до $1{,}8\cdot10^{308}$;

***Формат поля*** задает представление данных на экране или печати. Приняты форматы вида:

- ✓ *основной*;
- ✓ *денежный* с использованием символа валюты;
- ✓ *фиксированный* - два знака после запятой;
- ✓ *с разделителями разрядов* - запятая в качестве разделителя тысяч;
- ✓ *процентный* ;

- ✓ *экспоненциальный* - представление чисел с порядком, причем E заменяет порядок, например, 1,7E09 равносильно записи $1{,}7 \cdot 10^9$ .

Если был выбран тип поля ***Дата/Время***, то формат поля предлагается конкретно для этого типа. Для даты вы выбираете любой формат, но в дальнейшем данные будут вводиться только в нем. При попытке ввода данных в другом формате появится сообщение об ошибке.

***Количество десятичных знаков*** – количество цифр в дробной части

**Маска ввода**

Маска ввода применяется при вводе информации. Например, для ввода даты удобно создать маку типа _ _/_ _/ _ _, чтобы вводить только числа, не используя разделителей типа точки или знака /.

Маски ввода работают эффективно, если данные имеют небольшую длину и информативны. Маски ввода можно создавать только на поля типа даты или текстовых полей. Существуют стандартные маски ввода, но можно также создать свою собственную маску ввода, например, для ввода номера паспорта.

Удобно маску ввода создавать с помощью мастера. Для этого откройте нужную таблицу в режиме конструктора, щелкните на имени нужного поля. Затем внизу в области **Свойства поля** щелкните в строке Маска ввода, справа появится кнопка … , щелкните по ней. Появится диалоговое окно **Создание масок ввода** (рис. 6.2.2). Выберите нужную маску, щелкнув по ее имени и нажмите кнопку **Далее**. Если вы работали с датой, то на следующем шаге можно поменять заполнитель, **Готово**.

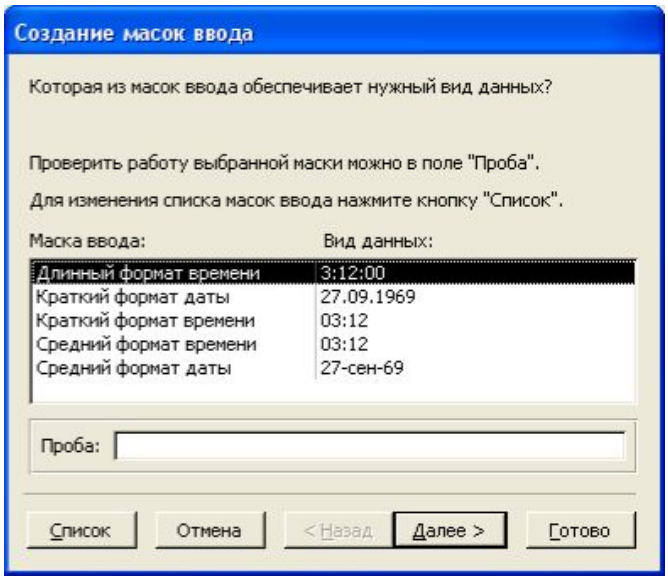



Рис. 6.2.2. Диалоговое окно создания маски ввода

Для создания маски вручную нужно использовать символы:

| *Символ* | *Описание* |
|---|---|
| 0 | Должна быть цифра от 0 до 9 |
| 9 | Должна быть цифра или пробел |

| | |
|---|---|
| L | Должна быть буква (A…Z или А…Я) |
| ? | Может быть буква |
| A | Должна быть буква или цифра |
| a | Может быть буква или цифра |
| # | Должна быть цифра, пробел, знак плюса (+) или минуса (-) |
| C | Произвольный символ |
| & | Должен быть любой символ или пробел |
| . , : ; / | Разделители |

В качестве примера напишем маску для ввода в базу данных номера паспорта, в котором есть буквенные и числовые части, каждая из которых имеет фиксированную длину и является обязательной:

*L L№000000*

**Подпись.**

Может пригодиться в работе. Она представляет собой текст, которым Access пользуется для маркировки поля в режиме таблицы, на формах и отчетах. Если не указывать значение подписи, то Access будет пользоваться для маркировки именем поля.

**Формирование условия на значение.**

Свойство **Условие на значение** располагается на вкладке **Общие**. Существует два момента в проверке вводимого значения – собственно название **Условие на значение** говорит само за себя и **Сообщение об ошибках** – это сообщение, которое выводится при несоответствии условия вводимому значению. Условие проверки значения работает эффективно при вводе числовых значений и данных типа Дата. Условие на значение, которое вводится в бланк запроса, является выражением, которое может быть сформировано с помощью **Построителя выражений**.

Используя, оператор AND (и), помните, что обе части условия проверки значения должны быть истинным для истинности условия в целом. При использовании OR (или) условие будет истинным, если истинной будет хотя бы одна часть выражения.

Например: <= 0 OR > =100

означает меньше или равно 0 или больше или равно 100.

**Обязательное поле и Индексированное поле.**

Также можно указать, является ввод значения в поле обязательным для каждой записи или нет (например, номер телефона не всегда присутствует в базе).

Индексирование поля позволяет ускорить поиск по нему, но требует дополнительного места для хранения информации. Если вы хотите индексировать поле, то можно разрешить повтор значений в нем, **установив Да (допускаются повторения)** или для уникального поля установив **Нет (Совпадения не допускаются)**.

**Выбор ключевых полей.**

Желательно (хотя и необязательно), чтобы в каждой таблице имелось ключевое поле. Оно используется для однозначной идентификации и упорядочения записей в таблице. Назначение ключевого поля рекомендуется при определении связей между данной таблицей и другими таблицами базы данных, поскольку ключевое поле автоматически индексируется с запретом совпадающих значений.

В большинстве случаев в качестве ключевого используется одно поле.

Чтобы сделать поле ключевым, щелкните в его строке в режиме конструктора и щелкните по кнопке **Ключевое поле** панели инструментов или выполните **Правка – Ключевое поле.**

**Свойства на вкладке Подстановка**

**Тип элемента управления**. Задаёт тип элемента управления, используемого по умолчанию для отображения данного поля в формах, отчётах и объектах в режиме таблицы. Для большинства полей выбирайте тип *Поле.* Если поле является чужим ключом, выберите *Список* или *Поле* со с*писком* для отображения значений из связанной таблицы. Эти же два типа элементов управления полезно использовать в случае когда поле должно содержать определённый список значений.

**Тип источника строк**. Если для свойства Тип элемента управления установлено значение *Список* или *Поле* со *списком*, это свойство позволяет задать тип источника данных для элемента управления: *Таблица/запрос, Список значений* или *Список полей*.

**Источник строк.** Если для свойства Тип источника строк установлено значение *Таблица/запрос,* укажите таблицу или запрос, из которого поступают значение для списка. Если источником данных служит список значений, введите значение для списка, разделяя их точкой с запятой.

Теперь на примере разберем создание таблиц базы данных.

Проанализировав поля, можно составить таблицу (табл. 6.2.1), которая поможет быстро создать таблицы базы данных и не забыть нужные свойства полей.

Таблица 6.2.1. Описание полей и их основных свойств

| ***Имя поля*** | ***Тип поля*** | ***Размер поля*** | ***Формат поля*** | ***Маска ввода*** | ***Условие на значение*** |
|---|---|---|---|---|---|
| Таблица ***Издательство*** | | | | | |
| КодИздательства | Числовой | | | | |
| Название | Текстовый | 25 | | | |
| Адрес | Текстовый | | | | |
| Телефон | Текстовый | | | 00a\-00\-00 | |

| ***Имя поля*** | ***Тип поля*** | ***Размер поля*** | ***Формат поля*** | ***Маска ввода*** | ***Условие на значение*** |
|---|---|---|---|---|---|
| Таблица ***Журналы*** | | | | | |
| КодЖурнала | Числовой | | | | |
| Название | Текстовый | 25 | | | |
| КодИздательства | Числовой | | | | |
| Цена | Денежный | | | | >0 |
| Таблица ***Пункт продажи*** | | | | | |
| КодПункта | Числовой | | | | |
| Адрес | Текстовый | | | | |
| ФИО | Текстовый | | | | |
| Таблица ***Продажи*** | | | | | |
| Дата продажи | Дата/Время | | Краткий формат даты | 00.00.0000;0;_ | |
| КодЖурнала | Числовой | | | | |
| КодПункта | Числовой | | | | |
| Количество | Числовой | | | | >=0 |

Вначале создадим таблицу ***Издательство***. Зайдем в создание с помощью конструктора и последовательно запишем названия полей, описанных в таблице 6.2.1 (рис.6.2.3). Затем в столбце **Тип данных** с помощью раскрывающегося списка выберем типы для полей в соответствии с таблицей 6.2.1. Для ввода номера телефона можно создать маску ввода. Для этого станьте в строке ***Телефон*** и в **Свойствах поля** в строке **Маска ввода** можно написать:

00a\-00\-00

(телефон может быть шестизначным или семизначным и разделяться знаком дефис «-»).

Также следует проанализировать, будут ли в таблице ключевые поля. Поле ***КодИздательства*** будет участвовать в связи двух таблиц ***Журналы*** и ***Издательство***, в этом поле не должно быть повторений. Поэтому в таблице это поле будет ключевым. Для установки ключевого поля установите курсор в строке ***КодИздательства*** и нажмите кнопку **Ключевое поле** на панели инструментов, слева от поля появится значок ключа. Теперь таблицу можно закрыть и в появившемся запросе задать название таблицы ***Издательство***.

Аналогично создаем таблицу ***Журналы*** и ***Пункт продажи*** в соответствии с таблицей 6.2.1.

Последняя таблица ***Продажи*** также создается в режиме конструктора. Заполните названия полей и типы в соответствии с таблицей 6.2.1. Для поля ***Дата продажи*** в свойствах поля можно выбрать формат поля **Краткий фор-**

**мат даты**, а также задать маску ввода с помощью мастера, установив курсор в строке **Маска ввода** и щелкнув по кнопке …

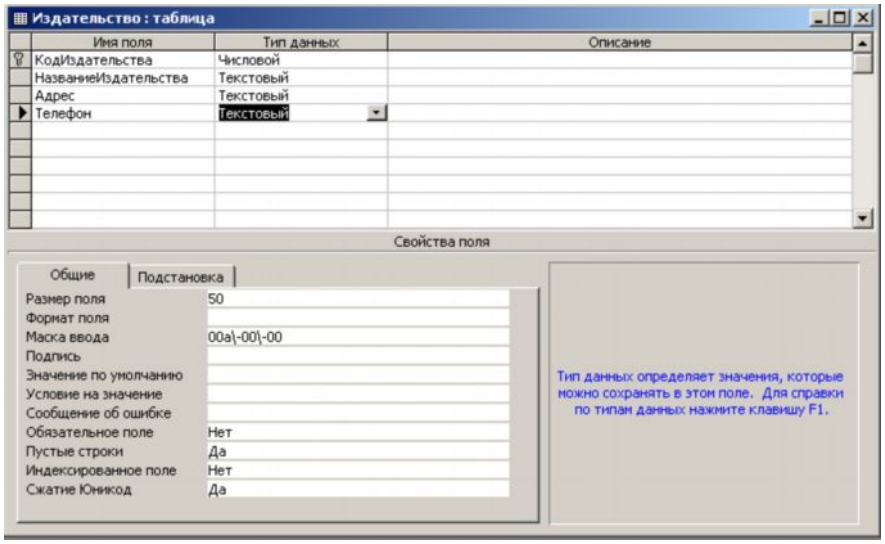



Рис. 6.2.3. Создание таблицы *Издательство* с помощью конструктора

Чтобы значения кодов журнала и кодов пунктов продажи можно было автоматически подставлять в таблицу **Продажи** из базовых таблиц, следует на вкладке **Подстановка** задать параметры в соответствии с рис.6.2.4.

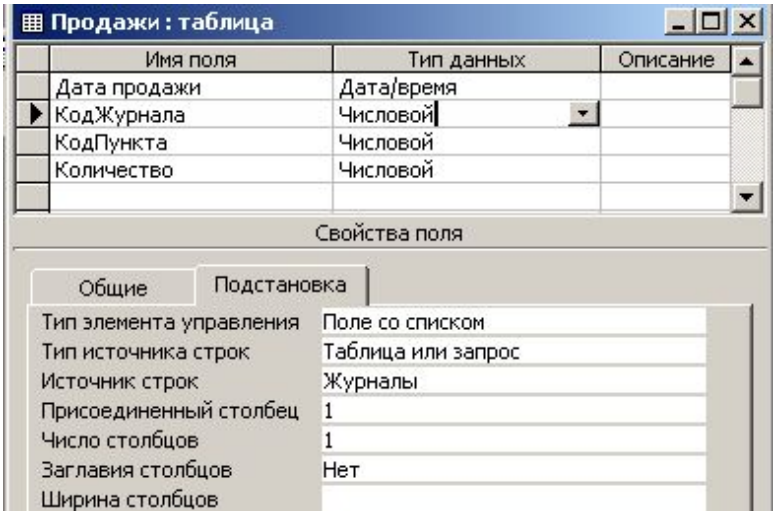
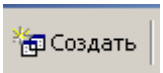



Рис. 6.2.4. Создание таблицы *Продажи* с помощью конструктора

Выберите тип элемента управления – поле со списком, а ниже в строке **Источник данных** выберите таблицу ***Журналы*** для поля ***КодЖурнала*** (или таблицу ***Пункт продажи*** для поля ***КодПункта***), номер присоединенного столбца – 1. В дальнейшем при заполнении данных в таблицу ***Продажи*** в полях ***КодЖурнала*** и ***КодПункта*** будут появляться кнопки раскрывающегося списка со значениями из базовых таблиц.

### 6.2.3. Создание таблиц в режиме таблицы

Щелкните по кнопке 

в верхней части окна базы данных на вкладке **Таблицы** и в открывшемся окне выберите **Режим таблицы.**

Появляется пустая таблица, по внешнему виду напоминающая электронную таблицу. Названия полей в созданной таблице – **Поле1**, **Поле2** и т.д. Чтобы переименовать столбец, следует выполнить двойной щелчок в шапке таблице в названии поля и задать новое имя или выполнить **Формат – Переименовать столбец**.

По умолчанию всем полям установлен тип данных – текстовый. Для изменения типа данных следует перейти в режим конструктора (**Вид – Конструктор**) и задать нужный тип (см. п.6.2.1.2).

В табличном режиме в последнюю строчку таблицы можно добавлять новые записи. Она помечена символом звездочки. Можно быстро перейти к последней строке, нажав кнопку **Новая запись** (рис.6.2.5).

При вводе данных в запись можно пользоваться щелчком мыши для выбора поля, в которое требуется ввести данные. Также можно использовать клавиатуру: клавиши TAB и Enter для перемещения слева направо по полям записи. Для перемещения на одно поле влево можно нажать **Shift+Tab**. В некоторых полях имеется раскрывающийся список, заданный мастером подстановок, для открытия списка следует щелкнуть по кнопке раскрывающегося списка или нажмите **Alt+**↓.

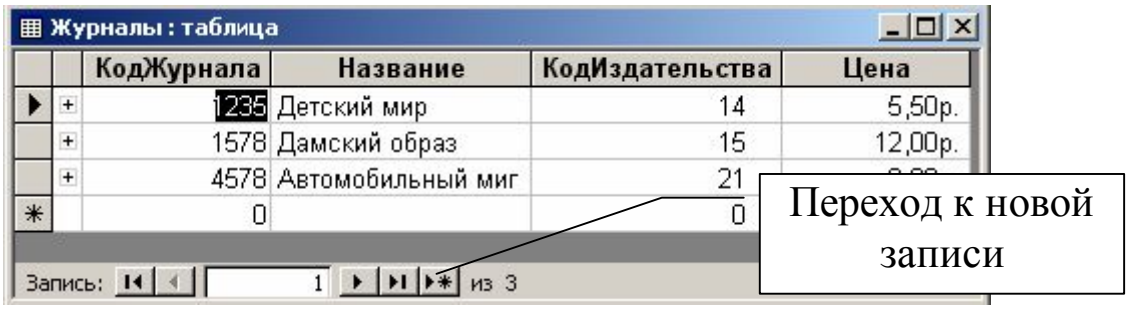



Рис. 6.2.5. Ввод данных в режиме таблицы

Для удаления столбца выполните **Правка – Удалить столбец**, для удаления записи: **Правка – Удалить запись**. ***Внимание***! Будет удален тот столбец или та запись, на которой стоял курсор.

Для добавления нового столбца выполните **Вставка – Столбец.**

Для добавления записи в середину таблицы выполните **Вставка – Запись**, для добавления записи в конец таблицы установите курсор в конец таблицы в запись, помеченную слева символом * и вводите данные.

Перемещение столбца: выделите столбец щелчком по заголовку столбца и перетащите его за этот заголовок в нужное место.

Для изменения ширины столбца следует установить указатель мыши на правый край заголовка столбца и перетащить, удерживая левую кнопку мыши вправо или влево. Аналогично можно изменить высоту строки.

6.2.4. Удаление таблиц

Осуществляется очень просто - на вкладке **Таблицы** выделите нужную таблицу и нажмите клавише **Delete**

## 6.3. Установка связей между таблицами

Сущность реляционной базы данных заключается именно в связях, которые устанавливаются между различными таблицами с данными. Для просмотра связей следует выполнить **Сервис – Схема данных** или нажать кнопку **Схема данных**  в правой части панели инструментов.

6.3.1. Типы связей

В зависимости от того, могут ли значения в поле связи повторяться по несколько раз в той или другой таблице, существуют такие типы связей (рис.6.3.1):

- ***Один к одному*** (взаимно-однозначный тип) – когда по обе стороны связи для любого значения в связующем поле имеется только одна запись;
- ***Один ко многим*** – когда по одну сторону связи для каких-то значений в связанном поле может быть несколько записей, а по другую – только одна;
- ***Многие ко многим*** – значения в полях неоднократно встречаются в записях той или другой связанных таблиц.

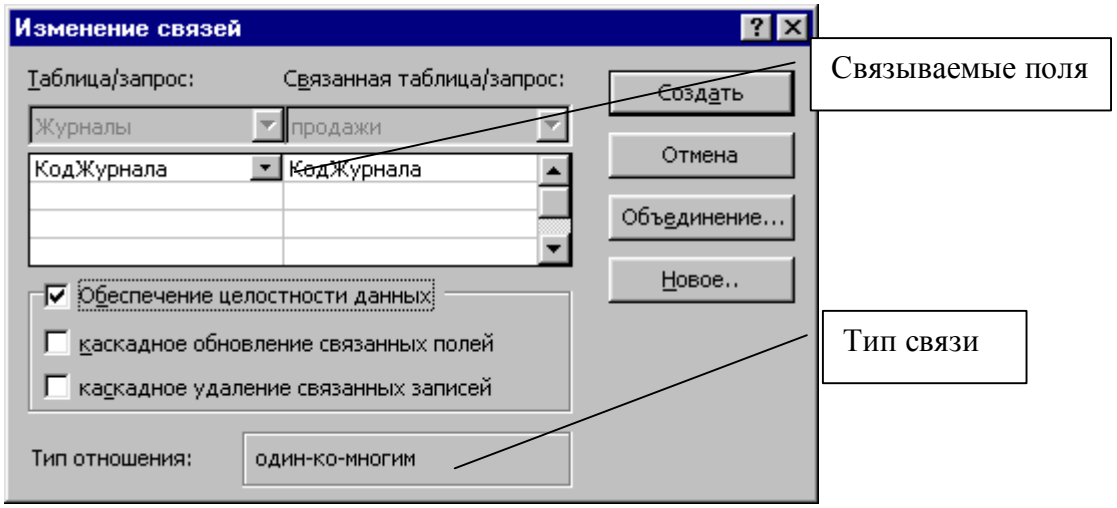



Рис. 6.3.1. Окно создания связи

Связи в базах данных довольно часто описываются в терминах **базовая/подчиненная таблица.** Связь создается парой полей, одно из которых находится в базовой таблице, а другое – в подчиненной; они могут содержать совпадающие значения. Когда значение в связанном поле записи базовой таб-

лицы совпадает со значением в связанном поле подчиненной, эти две записи называются связанными.

Access не требует жесткого соответствия между базовыми и подчиненными записями. Если не ограничить связи при их создании условием целостности данных, в структуре базы данных могут присутствовать базовые записи без подчиненных и подчиненные баз базовых.

### 6.3.2. Установка связей на схеме данных

Перед созданием связей следует закрыть все таблицы базы данных и открыть схему данных. При работе с окном **Схема данных** важно расположить таблицы таким образом, чтобы линии связей были четко видны и не пересекались друг с другом. При первом открытии этого окна появляется диалоговое окно **Добавление таблицы** (рис.6.3.2). Для добавления таблицы в схему данных следует выделить ее и щелкнуть по кнопке **Добавить** или щелкните по названию таблицы дважды левой кнопкой мыши. Если диалоговое окно добавления не появилось, выполните команду **Связи – Добавить таблицу** или щелкните правой кнопкой мыши по серой области схемы данных и из контекстного меню выберите эту же команду. После того, как будут присоединены все нужные таблицы, нажмите кнопку **Закрыть**.

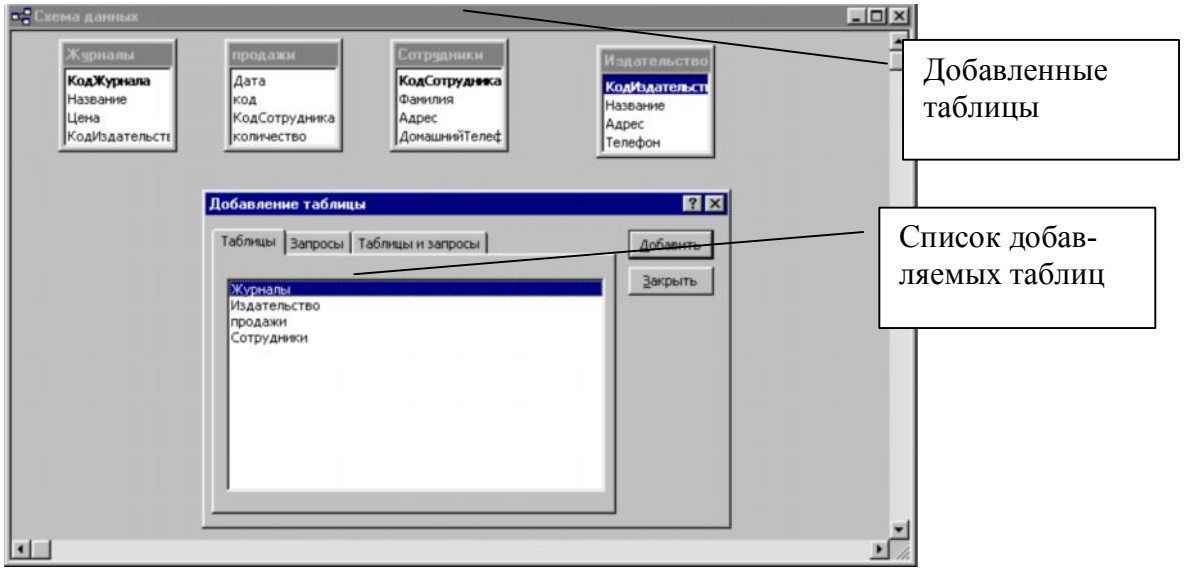



Рис. 6.3.2. Добавление таблицы в схему данных

Перемещаются окна таблиц на схеме данных также как и стандартные окна – подведите указатель мыши на заголовок окна и, удерживая левую кнопку мыши, переместите окно в нужном направлении. Если в правой части окна таблицы появилась полоса прокрутки, это означает, что отображаются не все имена полей. Можно увеличить размер окна, перетащив, удерживая левой кнопкой мыши, границу окна.

Для создания связи перетащите поле, используемое для связи, базовой таблицы на связываемое поле подчиненной таблицы, откроется окно, изображенное на рис. 6.3.1. В верхней части диалогового окна описываются поля, используемые для создания связи. Поле **Тип отношения**, расположение внизу окна, задает тип создаваемой связи. Если связанное поле базовой таблицы не индексируется, то Access ничего не знает о природе связи, и она описывается как «не определено». Если связанное поле базовой таблицы индексировано и в нем содержатся уникальные значения, то назначается тип связи «один-к-одному» или «один-ко-многим».

Чтобы потребовать обязательного наличия записи у каждой подчиненной записи, установите флажок **Обеспечение целостности данных**. При установленном флажке Access не позволит вам создать подчиненную запись, если в базовой таблице не существует ранее созданной записи с совпадающим значением в связанном поле.

Два других флажка **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей** доступны при установленном флажке обеспечения целостности данных и определяют, что происходит с записями подчиненной таблицы при внесении изменений в связанные с ними записи базовой таблицы.

## 6.4. Создание формы

Данные в таблицу вводить и редактировать намного удобнее, если воспользоваться экраном в виде бланка или формы. Такой способ ввода позволяет видеть на экране все данные одной записи и вводить дополнительный текст, поясняющий значение каждого поля.

В Access используется специальный мастер по созданию форм следующих видов:

- ✓ ***в один столбец*** – поля выводятся на экран в виде последовательности строк
- ✓ ***табличная форма*** – поля выводятся в виде строк и столбцов
- ✓ ***диаграмма*** – представляет числовые данные таблицы в виде диаграммы
- ✓ ***составная форма*** – объединяет в себе данные нескольких таблиц

Сформированная форма позволяет изменять, удалять, а также вводить новые данные в БД.

Создавать форму можно несколькими способами, но наиболее удобным является способ с помощью мастера. Для этого перейдите на вкладку **Формы** и щелкните дважды по **Создание формы с помощью мастера.**

Откроется диалоговое окно (рис. 6.4.1), в котором требуется выбрать таблицу или таблицы, для которых создается форма, а также поля из этих таблиц. Для выбора таблицы воспользуйтесь раскрывающимся списком **Таблицы и за-**

**просы**, внизу отобразится список поле этой таблицы. Для выбора одного какого-либо поля следует выделить его в списке **Доступные поля** и с помощью кнопки > перебросить его в область **Выбранные поля**. С помощью кнопки >> выбираются все доступные поля.

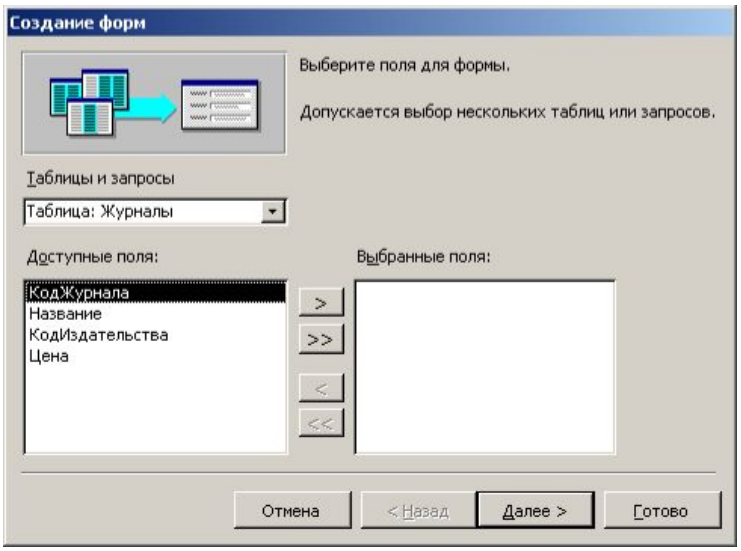



Рис. 6.4.1. Выбор полей при создании формы

Если выбрать поля из одной таблицы, то создается простая форма.

Если выбираются поля из двух связанных таблиц, то на следующем шаге можно выбрать вариант формы:

- ***Подчиненная форма*** – вверху отображаются записи базовой таблицы, а ниже записи подчиненной таблицы.
- ***Связанная форма*** – отображаются записи базовой таблицы, здесь же расположена кнопка, щелчком по которой можно открыть форму для работы с подчиненной таблицей.

Следующие шаги мастера создания формы – это задания внешнего вида формы (табличный, ленточный и т.п.), стиля оформления и имени форм.

На рис.6.4.2. представлена подчиненная форма для таблиц ***Журналы*** и ***Продажи***. Внизу формы размещаются строки с кнопками, с помощью которых можно перемещаться между записями. Для создания новой записи нажмите кнопку (**Новая запись**). Новая запись добавляется в конец базы данных.

Чтобы навсегда удалить запись из базы данных, отобразите ее в форме и нажмите кнопку (Удалить запись) на панели инструментов или выполните **Правка – Удалить запись,** при этом Access обязательно потребует подтвердить ваше решение.

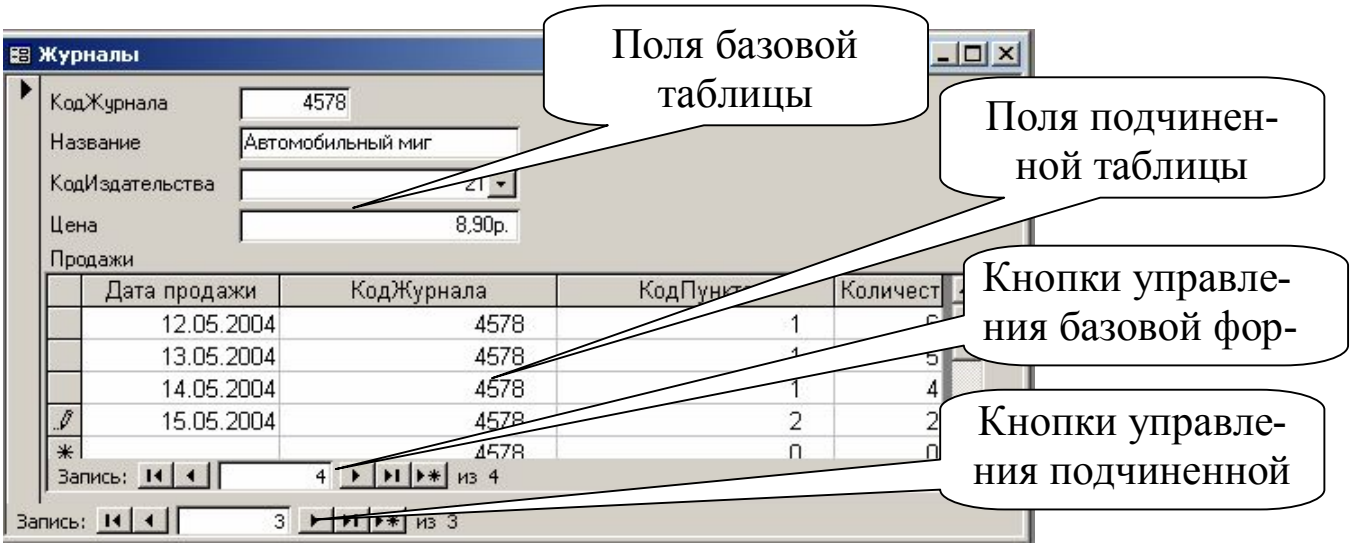



Рис. 6.4.2. Подчиненная форма

## 6.5. Просмотр данных в режиме таблицы

### 6.5.1. Поиск информации

Иногда требуется в таблице, содержащей много информации, найти нужную, – поле со значением, содержащим определенный фрагмент информации. Самый простой способ – открыть нужную таблицу и выполнить **Правка – Найти,** откроется диалоговое окно, изображенное на рис. 6.5.1. В текстовом поле **Образец** введите искомый фрагмент и нажмите кнопку **Найти далее**. Если такая информация будет найдена, то в таблице она подсвечивается черным выделением

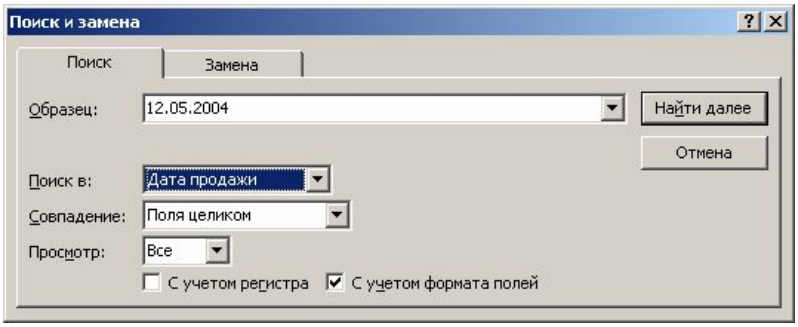



Рис.6.5.1. Диалоговое окно поиска информации в таблице

По умолчанию поиск осуществляется в том поле, в котором стоял курсор. Если вы хотите, чтобы Access произвел поиск во всех полях таблицы, следует в раскрывающемся списке **Поиск в** выбрать всю таблицу.

### 6.5.2. Сортировка информации

Самый простой способ сортировки – выделите столбец, по которому следует отсортировать таблицу, и щелкните по нужной кнопке  . В случае

числовых данных столбец отсортируется от меньшего значения к большему, или наоборот. В случае текстового поля сортировка осуществляется в алфавитном порядке.

Также можно выполнить команду **Записи – Сортировка – Сортировка по возрастанию (Сортировка по убыванию).**

## 6.6. Использование фильтров

Инструмент **Фильтр** позволяет отобрать в открытой таблице или форме данные, удовлетворяющие заданным условиям. Фильтры автоматически сохраняются при сохранении таблицы или формы. Таким образом, при повторном открытии таблицы можно повторно применить фильтр.

Чтобы применить фильтр, нажмите на кнопку **Применение фильтра**  на панели инструментов. Чтобы удалить фильтр нажмите эту же кнопку или выполните команду из меню **Записи**.

### 6.6.1. Фильтр по выделенному

Откройте таблицу или форму, с которой будете работать. Просматривая записи, найдите такую, которая содержит в нужном поле значение, которое должны содержать отобранные с помощью фильтра записи. Затем выделите это значение и нажмите кнопку **Фильтр по выделенному**  на панели инструментов или выполните **Записи – Фильтр – Фильтр по выделенному.** В любом случае список заменяется на меньший, в котором отображаются только записи с совпадением значения выделенного поля (рис.6.6.1).

Если перед применением фильтра по выделенному была выделена лишь часть значения поля, то отбор будет осуществляться только по этому критерию. Чтобы установить фильтр на основании нескольких значений одного и того же поля, найдите группу записей, в которой эти значения находятся рядом и выделите их (когда указатель мыши в ячейке примет вид ✞, щелкайте, удерживая клавишу Shift). Для фильтрации по значениям из нескольких полей следует разместить поля таким образом, чтобы поля с нужными значениями находились в смежных столбцах (при необходимости переместить столбцы), затем выделить данные, чтобы все они вошли в критерий отбора.

Все условия подчинены логическому **AND**.

Для снятия фильтра следует щелкнуть по кнопке удалить фильтр  на панели инструментов.

Затем при необходимости можно аналогично задать фильтрацию по выделенному по другому полю.

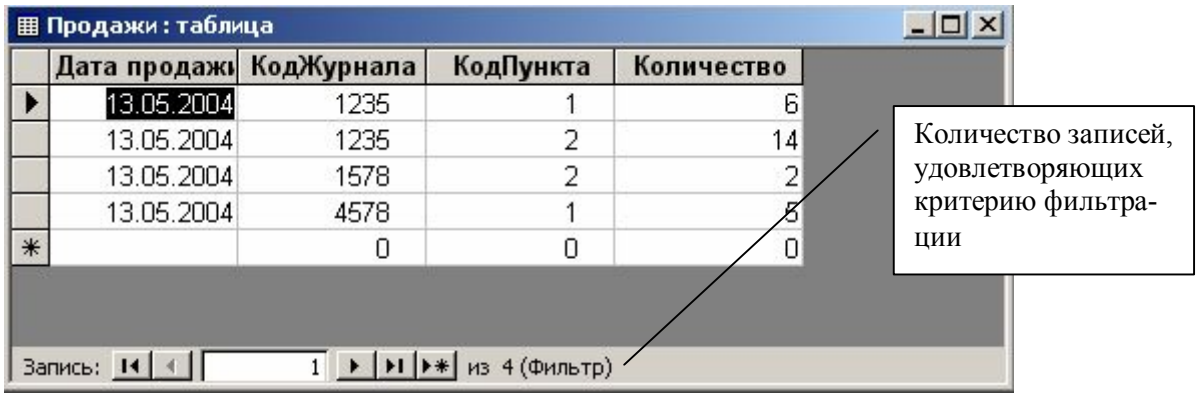



Рис. 6.6.1. Работа *Фильтра по выделенному*.

6.6.2. Обычный фильтр

Этот фильтр позволяет создавать условия по логическому **OR**. Для перехода в режим простого фильтра щелкните по кнопке **Изменить фильтр**  или выполните **Записи – Фильтр – Изменить фильтр**.

6.6.3. Фильтр «для»

Задается из контекстного меню текущего поля для значений этого поля (рис.6.6.2).

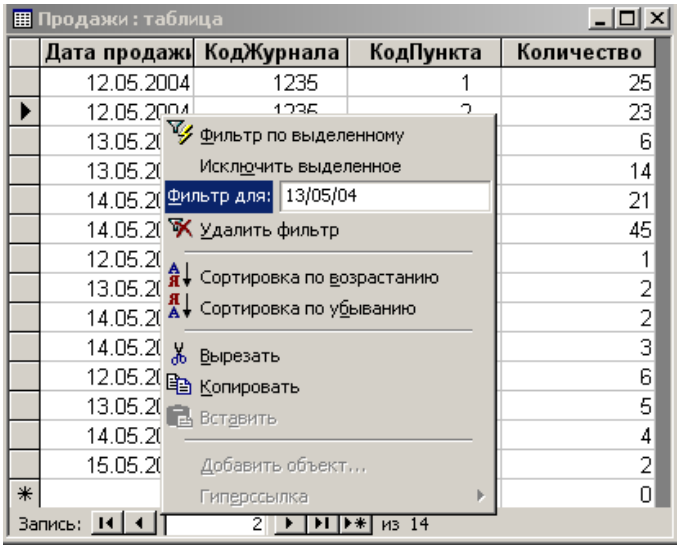



Рис. 6.6.2. Окно команды *Фильтр для.*

6.6.4. Расширенный фильтр

Этот фильтр является простейшим вариантом запроса. Выполните **Записи – Фильтр – Расширенный фильтр**, откроется окно, изображенное на рис.

6.6.3.

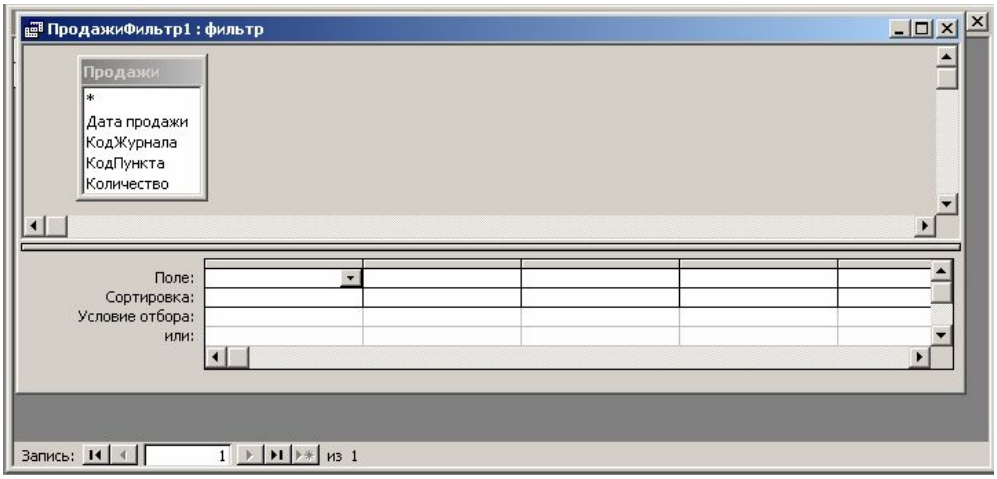



Рис. 6.6.3. Окно команды *Расширенный фильтр*

Далее следует задавать условия фильтрации, т.е. отсеивания ненужных записей (рис.6.6.4). Каждое условие состоит из частей:

- ***Поле***, по которому будут заданы искомые значения или другие условия отбора. Для добавления поля выполните двойной щелчок по его названию в списке, расположенном в верхней области конструктора. Также можно щелкнуть в пустом столбце в строке **Поле**, справа появится кнопка раскрывающегося списка, с помощь которой можно выбрать нужное поле.
- ***Условие отбора***. Для задания условия следует щелкнуть в строке **Условие отбора** нужного поля и указать его. Условие задается следующим образом: пишется оператор (=, <, >, <= или >=) и рядом указывается значение, например, >=50.

Пример задания условий отбора отображен на рис.6.6.4.

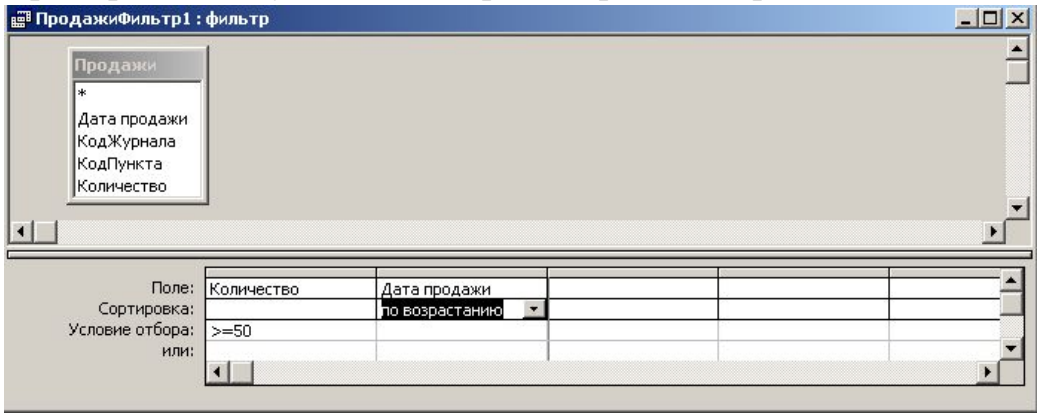



Рис. 6.6.4. Задание условий отбора с помощью *Расширенного фильтра*

Можно задавать условия отбора по нескольким полям, для этого заполните аналогично второй, третий и т.д. столбцы конструктора.

При отборе записей их также можно отсортировать, для этого щелкните в строке **Сортировка** соответствующего поля и из раскрывающегося списка вы-

берите вариант сортировки. При этом, если вы зададите сортировку по нескольким полям, то сортируются вначале записи по самому левому полю, затем по полю, расположенному в следующем столбце справа и т.д.

## 6.7. Создание запросов

Запросы – это средство Microsoft Access, позволяющее извлекать нужную информацию из базы данных, производить вычисления, используя данные из нескольких таблиц. С помощью запросов можно просматривать, анализировать, изменять данные в нескольких таблицах. Также запросы могут являться основой для создания форм или отчетов.

Запрос в Access хранится в виде SQL-выражения, которое описывает его и выполняет каждый раз при его запуске и поэтому является динамическим набором данных.

### 6.7.1. Запрос на выборку

Запросы представляют собой средства, с помощью которого можно отобрать записи, удовлетворяющие некоторым критериям. В результате запроса отображаются не все записи из таблицы или нескольких связанных таблиц, а только те, которые удовлетворяют запросу.

Запросы могут быть простыми (с одним условием отбора), сложными (производится ряд сравнений содержимого полей).

В основе любого запроса лежит **бланк запроса**, в строках и столбцах которого вводятся используемые поля и условия.

Результаты запроса всегда отображаются в режиме таблицы. Переключение между режимам конструирования запроса и табличным осуществляется нажатием кнопок на панели инструментов:

| | |
|---|---|
|  | Запуск запроса – отображение данных в табличном режиме |
|  | Переход в табличный режим |
|  | Переход в режим конструктора запроса |

Также можно выбрать соответствующий режим работы из меню **Вид**.

#### 6.7.1.1. Окно конструктора запросов

Для создания запроса в окне **Базы данных** перейти на вкладку **Запросы.**

Окно **Конструктора запросов** (рис.6.7.1) разделено на две части. В верхней части находятся окна таблиц со списками полей. Имя каждой таблицы отображается в строке заголовка такого окна.

В нижней части окна располагается бланк запроса, в котором выполняется вся работа. Каждый столбец бланка представляет одно поле, используемое в запросе. При составлении запроса на основе нескольких таблиц между ними необходимо установить связь.

Первая строка бланка запроса служит для задания полей. Это могут быть поля, которые должны попасть в набор записей, и поля, используемые только для задания условий сортировки или отбора данных из таблицы. Имя таблицы отображается во второй строке бланка запроса. Третья строка бланка позволяет задать порядок сортировки для выбранного поля.

Флажки в строке бланка **Вывод на экран** отвечает за включение полей в набор записей запроса.

Добавить нужные поля в бланк запроса можно путем перетаскивания их имён из списка, находящегося в верхней части окна конструктора в строку бланка **Поле** или двойным щелчком на имени поля.

Для удаления поля из запроса следует выделить колонку и нажать **DEL** или выполнить **Правка- Удалить столбцы.**

### 6.7.1.2. Создание условий отбора

С помощью задания условий можно отобразить не все записи, а только те, которые удовлетворяют установленным критериям отбора.

Для создания условия следует в бланке запроса щелкнуть в строке **Условие отбора** того столба, который содержит название поля, по которому осуществляется отбор. Затем введите оператор вместе со значением, например, >50. Условия можно комбинировать, для этого следует выбрать несколько полей и на нужном поле в строке **Условие отбора** задать соответствующее условие.

Рассмотрим несколько примеров задания условий. Если требуется совпадение в числовом, текстовом или другом поле, то в условии отбора следует написать это число, текст или дату (рис.6.7.1). Если следует отобразить записи для двух вариантов значения поля, то первое условие следует вписать в строке **Условие отбора:**, а второе – в строке **или:**. Для задания условия сравнения (больше, меньше, больше или равно, меньше или равно) используются операторы сравнения >, <, >=, <=.

При задания условия по полю типа **Дата** дату с условием отбора следует указать в символах **#.** При задании условия по текстовому полю в условии отбора текст берется в кавычки. Для задания условия по логическому полю в условие отбора нужно написать **Да** или **Нет**.

Часто приходится использовать для отбора одно и тоже поле, но с разными условиями. Например, необходимо отобразить сведения о продажах между 1 января 2004 г. и 1 февраля 2004 г. В этом случае следует использовать опера-

тор логического умножения And (и). условие отбора будет выглядеть как запись двух условий и между ними And:

***>=#01.01/04# And <=#01.02.04#***

Примеры задания условий на значение текстового поля

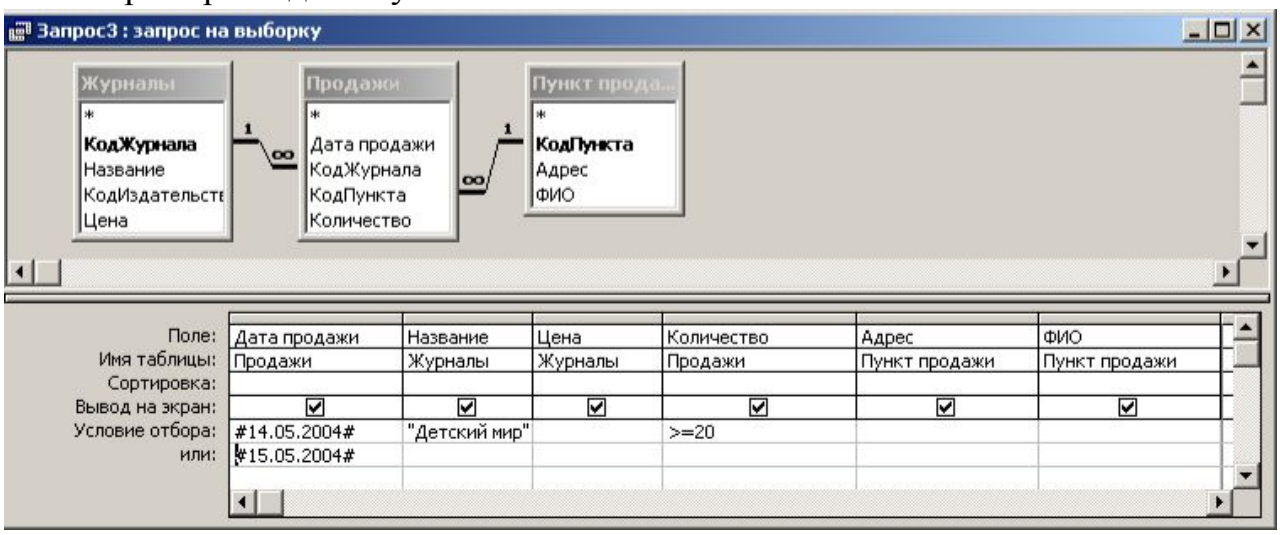



Рис. 6.7.1. Задание условий отбора в бланке запроса

| ***Выражение*** | ***Описание*** |
|---|---|
| “Детский мир” | Отображение сведений о журнале с названием «Детский мир» |
| “Детский мир” or “Мурзилка” | Отображение сведений о журналах с названием «Детский мир» или «Мурзилка» |
| In (“Детский мир”,” Мурзилка ”) | Отображение сведений о журналах с названием «Детский мир» или «Мурзилка» |
| Not “Детский мир” | Отображение сведений о всех журналах, кроме журнала «Детский мир» |
| < “Г” | Названия начинаются с букв, находящихся в диапазоне с А до Д |

Для задания условий отбора на поле типа Дата можно использовать функции работы с датой, которые описаны ниже в разделе 6.7.2. Примеры задания условий на значение поля формата Дата/Время:

| ***Выражение*** | ***Описание*** |
|---|---|
| #01.12.04# | Отображение сведений о продажах за дату 01.12.04 |
| Between #01.12.04# And #10.12.04# | Использование оператора **Between...And** для отображения сведений о продажах не ранее 1-дек-04 и не позднее 10-дек-04. |
| #01.12.04# Or #2.12.04# | Отображение сведений о продажах за 1-дек-04 или 2-дек-04. |
| > Date ( ) -10 | Сведения о продажах за последние 10 дней |

При работе с текстовыми полями в условии отбора можно использовать оператор **Like** – поиск текста или части текста в значении поля.

Примеры оператора Like для поля название журнала.

| ***Оператор*** | ***Назначение*** |
|---|---|
| Like ″Д*″ | Отбор журналов, начинающихся на букву Д |
| Like ″*мир″ | Отбор журналов, название которых заканчивается на *мир* |
| Like ″*мир*″ | Отбор журналов, в названии которых встречается слово на *мир* |
| Like "[А-К]*" | Отбор журналов, начинающихся с букв от А до К |

Также оператор **Like** можно использовать для задания условия по полю типа Дата. Например,

запись **Like "*.06.*"** – означает выбор всех дат за 6-й месяц,

**Like "*.06.2007"** – выбор всех дат за 6-й месяц 2007 года (количество цифр года должно указываться одинаково с выводом даты в таблице).

Условие отбора можно задавать с помощью построителя выражения, который рассмотрен ниже в разделе 6.7.2.

6.7.1.3. Параметрический запрос

Представляет собой вариант базового запроса на выборку.

В режиме конструктора выбираются таблицы и нужные поля из них. Поле строки **Условие отбора** в соответствующем столбце заполняется не конкретным значением отбора, а обращением к пользователю. Это обращение должно быть заключено в квадратные скобки, например, [*Введите название журнала* ], [*За какую дату хотите посмотреть данные*]. ***Замечание***: текст внутри скобок не должен совпадать с названием поля.

Также параметрический запрос можно использовать вместе с оператором Like. Например, если требуется отобрать только журналы, начинающиеся на заданную букву, которая будет вводится. В этом случае оператор Like будет выглядеть:

***Like [Введите первую букву наименования журнала] & ″*″***

Знак & обозначает объединение строк – в данном случае первую букву (введенную) с любым текстом (″*″).

6.7.2. Вычисления в запросах

Вычисляемое поле может быть создано в запросе, форме или отчете.

Чтобы создать вычисляемое поле, щелкните в бланке запроса в новом столбце в строке **Поле** и введите выражение для расчета. В выражении название поля нужно заключать в квадратные скобки.

По умолчанию новым вычисляемым полям присваивается имя **Выражение1**, **Выражение2** и т.д + **:** (двоеточие). Имя поля можно поменять на свое усмотрение.

*Пример 1.* Для расчета суммы от продажи следует создать новое расчетное поле, в строке **Поле** записать:

***Сумма продажи: [Цена]*[Количество]***

*Пример 2.* Определить новую цену журнала со скидкой 5% :

**Новая цена: [Журналы]![Цена]*0,95**

где в имени поля вначале перед восклицательным знаком указывается название таблицы, откуда берется поле для расчета.

Удобно создавать вычисляемые поля с помощью **построителя выражения**, который вызывается нажатием кнопки  на панели инструментов. Откроется диалоговое окно, изображенное на рис.6.7.2.

В верхней части построителя вводится текст формулы. Ниже есть кнопки с математическими операторами(умножить, разделить и т.п.). Нижняя часть построителя разделена на три части. В левой части выбирается объект базы данных (таблица, запрос) или функции (встроенные функции). Для раскрытия группы следует дважды щелкнуть по значку папки с символом +, расположенному слева от группы. Если слева выделить таблицу или запрос, то в средней части появляется соответственно список доступных полей таблицы. Если слева выделяется группа **Встроенные функции,** то в средней части отображается список категорий функций (математические, дата/время и т.д.), при выделении категории функций в правой части отображается список функций их этой категории.

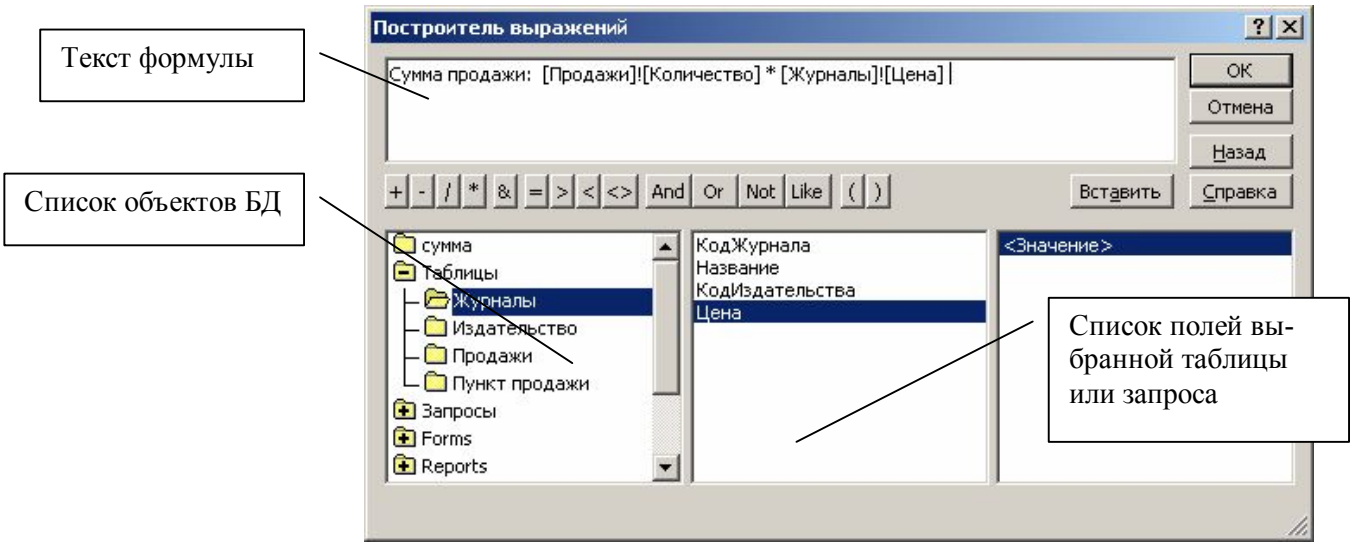



Рис. 6.7.2. Диалоговое окно построителя выражений

Для добавления поля в текст формулы следует дважды щелкнуть по его названию в средней части построителя, при этом ссылка на поле состоит из

имени таблицы, символа восклицательного знака и имени поля.

В вычисления также можно включать и слова. Например, чтобы объединить Фамилию и Имя в одно поле следует написать:

*[Фамилия] & " " & [Имя],*

где знак & обозначает конкатенацию (объединение) строк, а между фамилией и именем вставляется пробел для отделения их друг от друга.

Для работы с датами можно использовать следующие функции:

| ***Функция*** | ***Описание*** |
|---|---|
| Year (дата) | Возвращает значение года |
| Month (дата) | Возвращает значение месяца |
| Day (дата) | Возвращает значение дня |
| WeekDay (дата) | Возвращает номер дня недели |
| Date( ) | Возвращает текущую системную дату |
| DateDiff (интервал;дата1; дата2) | Определяет количество временных периодов между двумя датами |
| DateAdd (интервал; n; дата) | Вычисляет новую дату, отстоящую на n временных интервалов от указанной даты |
| DateDiff (интервал; дата1; дата2) | Вычисляет число временных интервалов между двумя датами |

Значение параметра ***Интервал*** в функциях принимает значения:

| | |
|---|---|
| yyyy | Год |
| q | Квартал |
| m | Месяц |
| y | Day of year |
| d | День |

| | |
|---|---|
| w | День недели |
| ww | Неделя |
| h | Часы |
| n | Минуты |
| s | Секунды |

В режиме конструктора у вычисляемого поля можно изменить его свойства, например, можно округлить значение вычисляемого поля, изменив в окне его свойств число десятичных знаков. Для открытия окна свойств поля (рис.6.7.3) щелкните в названии поля и выполните **Вид – Свойства** или щелкните правой кнопкой мыши и из контекстного меню выберите **Свойства**.

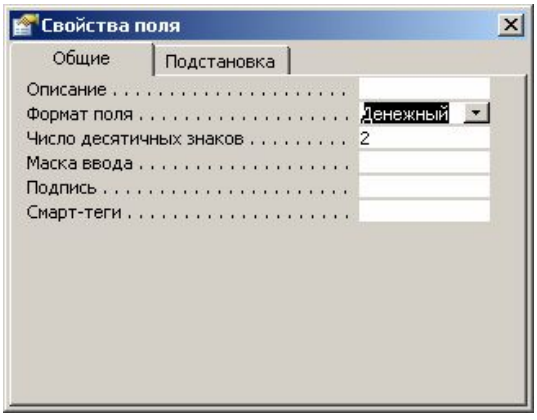



Рис. 6.7.3. Диалоговое окно *Свойств поля*

6.7.3. Запрос с выводом набора значений

С помощью этого вида запроса можно просмотреть максимальные или минимальные значения поля.

Для создания такого запроса выполните:

1. В режиме конструктора выберите таблицу и поля. В нашем примере находятся два самых дорогих журнала, поэтому выбрана таблица **Журналы** и поля из этой таблицы (рис.6.7.4).

2. Отсортируйте записи по нужному полю. Для просмотра минимальных значений таблица должна быть отсортирована по возрастанию, для просмотра максимальных – по убыванию. В нашем примере произведена сортировка по убыванию по полю ***Цена***.

3. В раскрывающемся списке **Набор значений** на панели инструментов выберите подходящее количество максимальных или минимальных значений. Если нужно число отсутствует в списке, то его следует ввести вручную.

4. Для просмотра результатов запроса выполните любое из действий, описанных в п. 6.7.1

***Замечание***. Если в таблице несколько повторяющихся максимальных или минимальных значений поля, то при просмотре результатов будет соответственно больше записей.

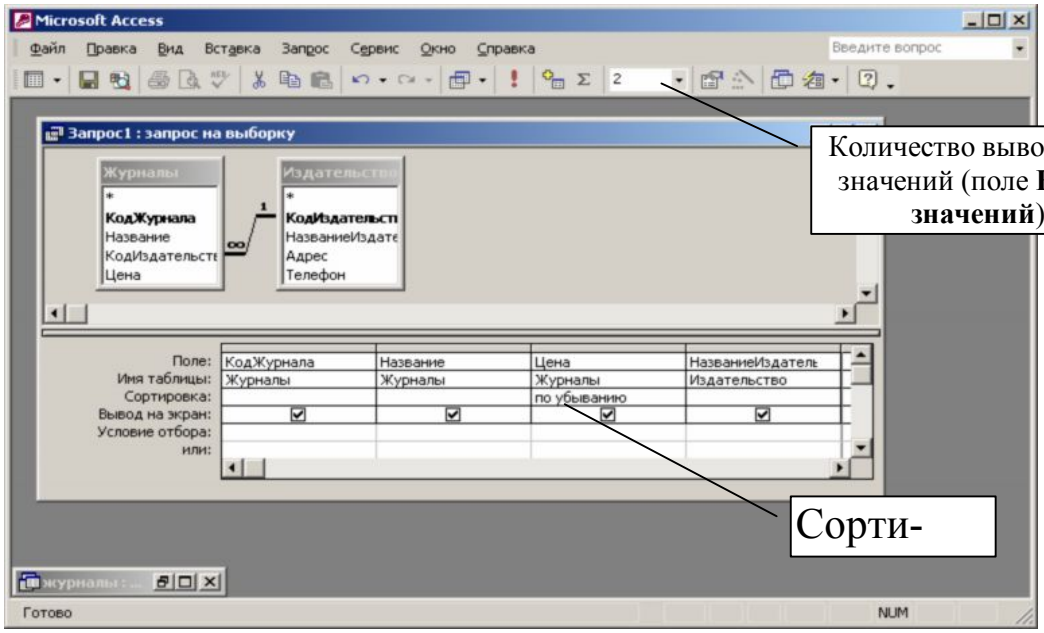



Рис. 6.7.4. Создание запроса с набором значений

6.7.4. Запрос с подведением итогов по записям

Этот вид запроса позволяет обобщить информацию с помощью строки **Групповая операция**. Для включения в бланк запроса этой строки выполните

**Вид – Групповые операции** или нажмите на кнопку Σ .

Для работы со строкой **Групповая операция** в бланке запроса должно быть хотя бы одно поле, которое содержит информацию для подведения итога – обычно это числовое или денежное поле. Для этого поля следует в строке **Групповая операция** из списка выбрать функцию для вычисления, описание функций приведено в таблице ниже. Если в таблице нет ни одного поля для обобщения, то список функций будет состоять только из одной функции **Count**.

| ***Функция*** | ***Описание*** |
|---|---|
| *SUM* | Суммирование значений группы |
| *AVG* | Вычисление среднего арифметического значения группы |
| *MIN* | Определение минимального значения в группе |
| *MAX* | Определение максимального значения в группе |
| *COUNT* | Вычисление количества значений в группе |
| *STDEV* | Стандартное отклонение для выборки, состоящей из значений группы |
| *VAR* | Дисперсия для выборки, состоящей из значений группы |
| *FIRST* | Первое значение в группе |
| *LAST* | Последнее значение в группе |

Например, для определения минимальной цены журнала бланк запроса будет выглядеть следующим образом (рис.6.7.5):

Также такой запрос можно использовать для подведения итогов по группам. Для этого в бланк запроса следует добавить поле, которое будет применяться для группировки записей (обычно в этом поле имеются повторяющиеся значения), и поле для подведения итогов с заданием функции вычисления.

Например, требуется по каждой дате найти общее количество каждого проданного журнала. Бланк такого запроса представлен на рис.6.7.6.

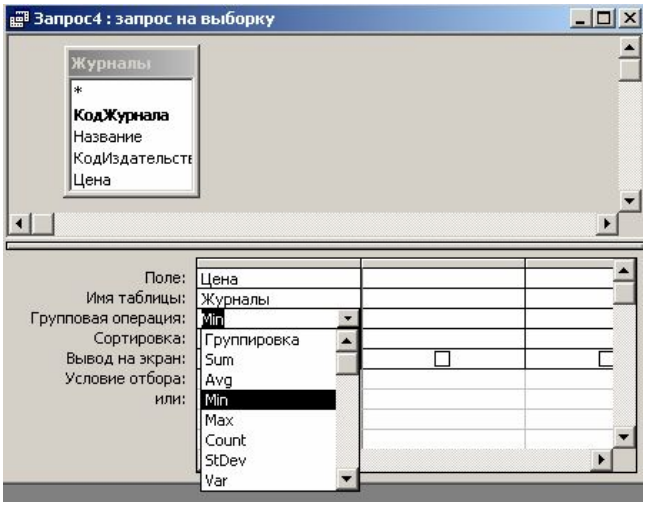



Рис. 6.7.5. Создание запроса определения минимальной цены журнала

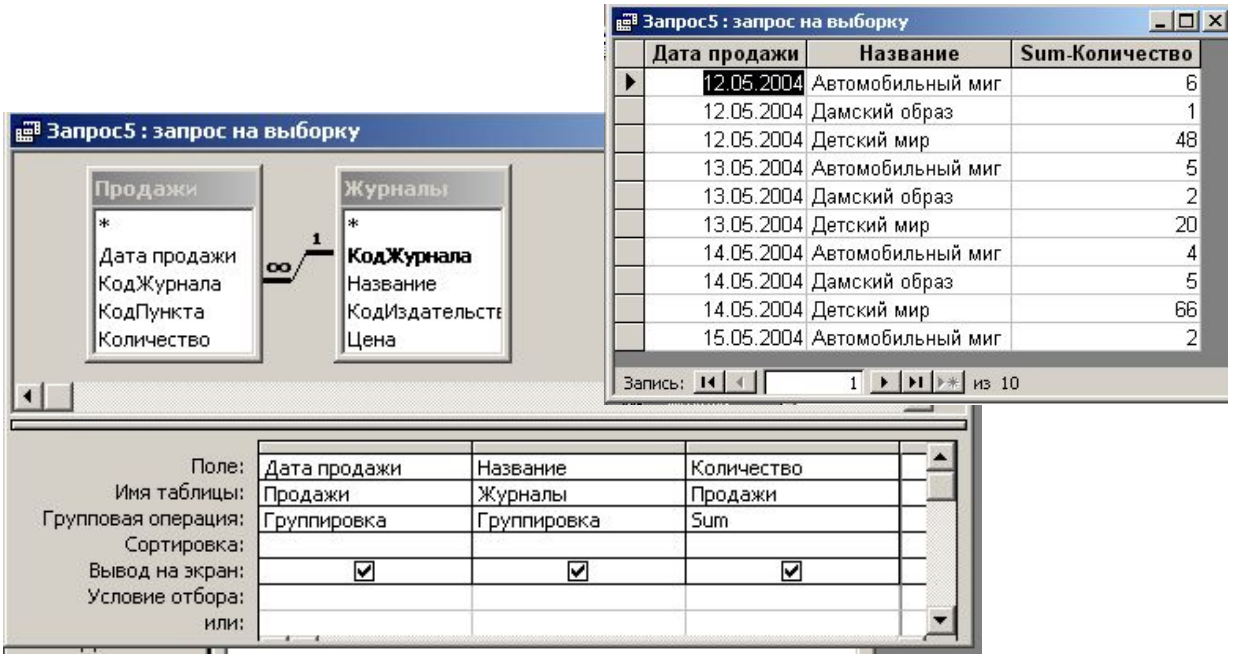



Рис. 6.7.6. Бланк запроса и результаты запроса подведения итогов по продаже журналов по датам

### 6.7.5. Перекрестные запросы

Перекрестный запрос является разновидностью итогового запроса. С помощью перекрестного запроса можно увидеть вычисляемые значения в виде таблицы, напоминающую электронную. Он позволяет компактно отображать данные и объединять однородную информацию.

Для создания перекрестного запроса лучше воспользоваться мастером. Для его вызова следует на вкладке **Запросы** щелкнуть по кнопке **Создать** и в открывшемся окне выбрать **Перекрестный запрос**.

Вначале перед созданием перекрестного запроса нужно проанализировать, расположены ли данные в одной таблице. если все нужные поля размещены в одной таблице, то на первом шаге (рис.6.7.7) в качестве основы следует установить переключатель на **Таблицы** и выделить нужную таблицу. Если поля находятся в разных таблицах, то вначале следует создать новый запрос на выборку, в который поместить нужные поля, и на первом шаге мастера выбирается за основу ранее созданный запрос.

На втором шаге выбирается от одного до трех полей, которые будут служить заголовками строк и содержат информацию для группировки значений, обычно это поля, содержащие повторяющуюся информацию. Далее аналогично выбирается поле, значений которого будут использоваться в качестве заголовков столбцов. И только затем выбирается поле с обобщаемой информацией и функция для подведения итогов. Кроме того, можно указать, нужно или нет включать итоговые значения для каждой строки.

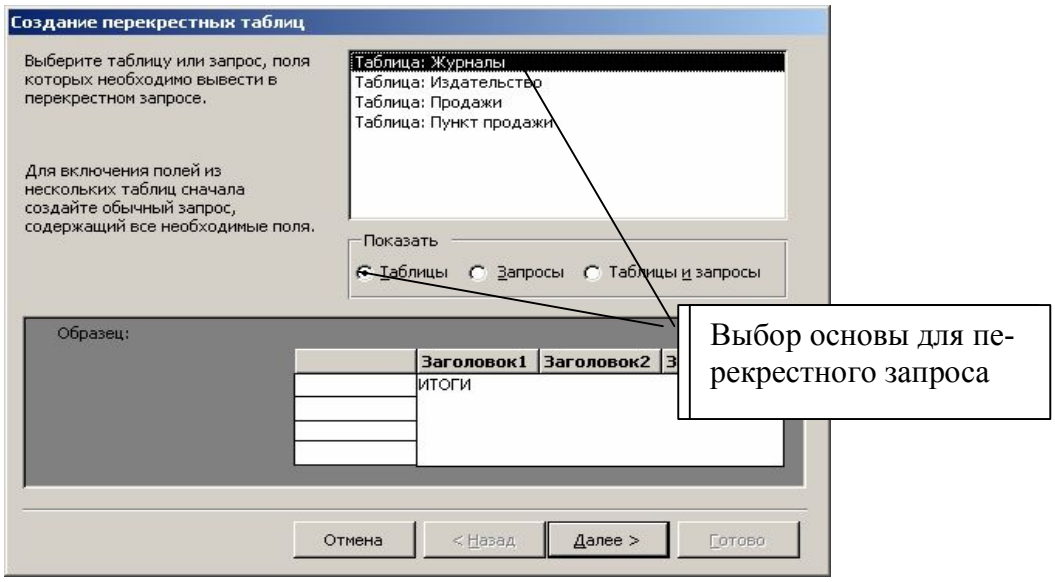



Рис. 6.7.7. Создание перекрестного запроса

Для примера построено два перекрестных запроса. Вначале был создан запрос на выборку, в который поместили поля из таблиц ***Дата продажи***, ***Адрес*** торговой точки, ***Название*** журнала и ***Количество***. В первом запросе (рис.6.7.8, вверху) в качестве заголовков строк было выбрано поле дата, в качестве заголовков столбцов – поле название журнала, а для поля количество выбрана функция для вычисления **Сумма**. Второй запрос (рис.6.7.8, внизу) отличается только тем, что в качестве заголовков строк выбрано два поля: дата продажи и адрес торговой точки.

Запрос2_перекрестный : перекрестный запрос

| Дата продажи | Итоговое значение Количество | Автомобильный миг | Дамский образ | Детский мир |
|---|---|---|---|---|
| 12.05.2004 | 55 | 6 | 1 | 48 |
| 13.05.2004 | 27 | 5 | 2 | 20 |
| 14.05.2004 | 75 | 4 | 5 | 66 |
| 15.05.2004 | 2 | 2 | | |

Запись: 1 из 4

Запрос2_перекрестный1 : перекрестный запрос

| Дата продажи | Адрес | Итоговое значение Количество | Автомобильный миг | Дамский образ | Детский мир |
|---|---|---|---|---|---|
| 12.05.2004 | Донецк, Артема, 156 | 32 | 6 | 1 | 25 |
| 12.05.2004 | Донецк, Ткаченко, 78 | 23 | | | 23 |
| 13.05.2004 | Донецк, Артема, 156 | 11 | 5 | | 6 |
| 13.05.2004 | Донецк, Ткаченко, 78 | 16 | | 2 | 14 |
| 14.05.2004 | Донецк, Артема, 156 | 51 | 4 | 2 | 45 |
| 14.05.2004 | Донецк, Ткаченко, 78 | 24 | | 3 | 21 |
| 15.05.2004 | Донецк, Ткаченко, 78 | 2 | 2 | | |

Запись: 7 из 7

Рис. 6.7.8 Перекрестные запросы

## 6.8. Создание отчета

Отчеты применяются для отображения информации из базы данных и для вывода ее на принтер.

***Замечание***. Если на компьютере не настроен принтер, то просмотр отчета будет невозможен. Также перед созданием отчета следует проанализировать, какие данные в нем будут представлены. Поэтому иногда требуется вначале создать запрос, а затем на его основе создавать отчет.

Для создания отчета перейдите на вкладку **Отчеты**.

Быстрый способ создания простого отчета – создание ***автоотчета***. Для его вызова щелкните по кнопке **Создать** и выберите:

- ***Автоотчет: в столбец*** – напоминает обычную форму, каждое поле выводится в отдельной строке, слева расположено название поля, а справа – его содержимое. Такой отчет может растянуться на несколько страниц
- ***Автоотчет: ленточный*** – информация располагается в виде таблицы

В этом же диалоговом окне ниже следует выбрать таблицу или запрос, по которому строится отчет. После нажатия кнопки **Ок** сразу создается готовый отчет. Недостатком такого типа отчета является невозможность добавления в отчет дополнительных параметров (сортировки, группировки и т.п.).

В большинстве случаев рекомендуется создавать отчет с помощью Мастера отчетов. После нажатия кнопки **Создать** выберите вариант **Мастер отчетов** и с помощью кнопки **Далее** перейдите на следующий шаг (рис.6.8.1). Здесь с помощью раскрывающегося списка **Таблицы и запросы** следует выбрать источник данных для отчета, а ниже выбираются поля, которые войдут в отчет. Поля можно выбирать из нескольких таблиц. Для переноса одного поля в область **Выбранные поля** следует либо дважды по нему щелкнуть, либо нажать кнопку **>**. Чтобы перенести все поля, нажмите кнопку **>>**. После выбора полей нажмите кнопку **Далее**.

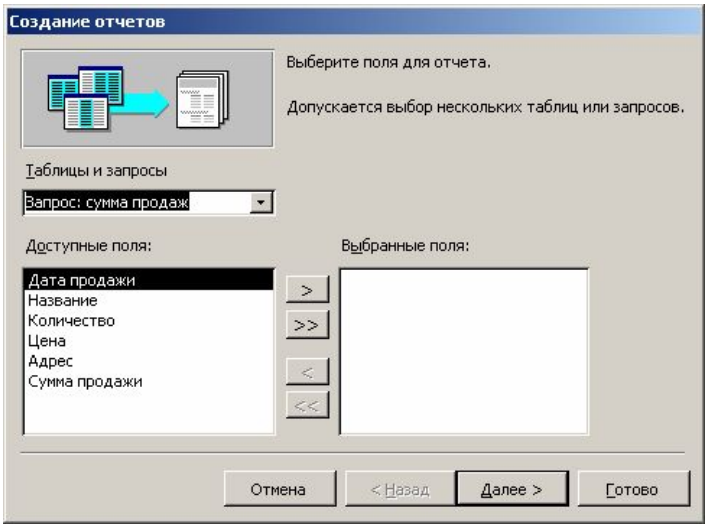



Рис. 6.8.1. Выбор используемых полей для отчета

На следующем шаге (рис.6.8.2) предлагается задать структуру отчета с помощью группировки информации. Иногда Access сам предлагает поле для группировки. Если такого не произошло, то группировку можно задать самостоятельно – выделить поле, содержащее повторяющиеся значения, и с помощью кнопки > перебросить его в образец. Можно задавать несколько уровней группировки. На этом шаге также можно задать параметры группировки. Например, если вам не подходит, что предложена группировка по месяцам даты, то с помощью кнопки **Группировка** можно изменить интервал группировки (по датам, неделям и т.п.).

После выбора полей группировки щелкните по кнопке **Далее** для перехода на следующее диалоговое окно. На следующем шаге можно произвести сортировку. Для этого в области с номером **1** с помощью раскрывающегося списка выберите поле, по которому следует отсортировать записи и щелчком по кнопке **по возрастанию/по убыванию** задайте порядок сортировки. В области **2** можно задать вторичный критерий сортировки. В этом же окне имеется кнопка **Итоги**, щелчком по которой открывается новое диалоговое окно. Это окно позволяет подвести итоги по числовым полям. Чтобы включить в отчет итог, установите флажок (∨) на пересечении поля и функции. Также можно установить флажок **Вычислить проценты**, при этом общая сумма будет принята за 100%, а по каждой группе выведется доля в процентном отношении.

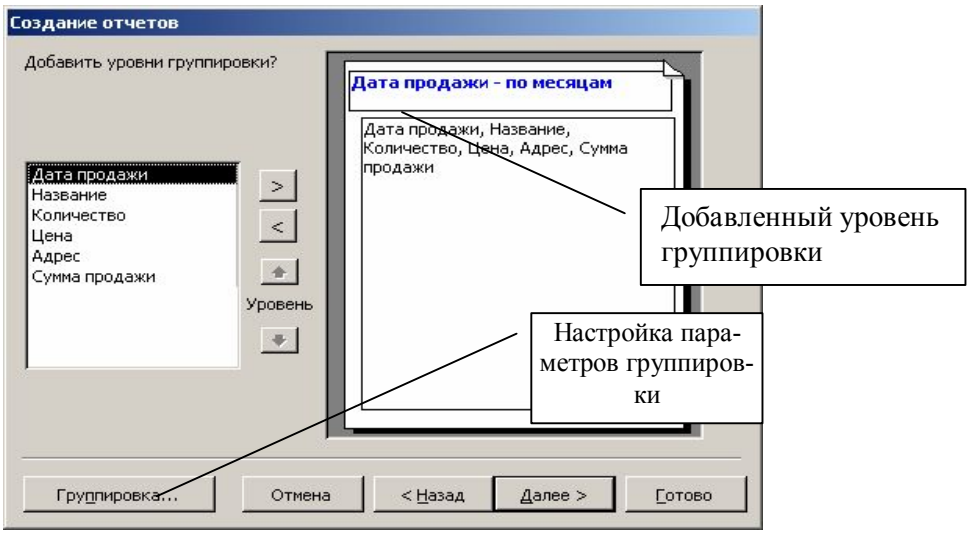



Рис. 6.8.2. Добавление уровней группировки при создании отчета

После щелчка по кнопке **Ок** и возврата в окно с сортировкой щелкните по кнопке **Далее**. Следующий шаг позволяет задать макет оформления отчета: ступенчатый, блок, структура 1, структура 2, по левому краю 1, по левому краю 2. Вид отчета будет зависеть от того, как была определена группировка записей. Здесь же следует выбрать ориентацию страничного листа (книжная или альбомная).

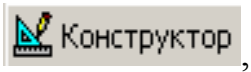
Следующий шаг мастера – выбор стиля оформления отчета.

На последнем шаге следует указать имя создаваемого отчета (по умолчанию названием отчета предлагается название таблицы, на основе которой он создавался).

Если выбран вариант **Просмотреть отчет**, то после нажатия кнопки **Готово** на экране в режиме просмотра отображается готовый отчет. В нижней части отчета имеется область с кнопками управления просмотром, там отображается номер текущей страницы, общее количество страниц и кнопки перехода.

## 6.9. Модификация и форматирование форм и отчетов

Под модификацией формы или отчета понимается изменение размеров элементов, не помещающихся на форме или отчете, изменение местоположения элементов, добавление элементов, вычислений. Часто после создания отчета в него требуется внести изменения, например, Access очень часто не может разместить имя или значение поля по ширине столбца и обрезает буквы. Такие проблемы решаются в основном изменением размера шрифта или простым редактированием текста заголовка.

Это выполняется в режиме конструктора. Чтобы начать работу с макетом отчета или формы, необходимо открыть его в режиме конструктора. Для этого можно перейти на вкладку **Отчеты** (или **Формы**), выделить нужный отчет (или форму) и щелкнуть по кнопке 

, или открыть отчет для просмотра и перейти в режим конструктора с помощью команды **Вид – Конструктор**.

Форма в режиме конструктора имеет вид, представленный на рис. 6.9.1. Окно состоит из зон ***Заголовок формы***, ***Область данных*** и ***Примечание***.

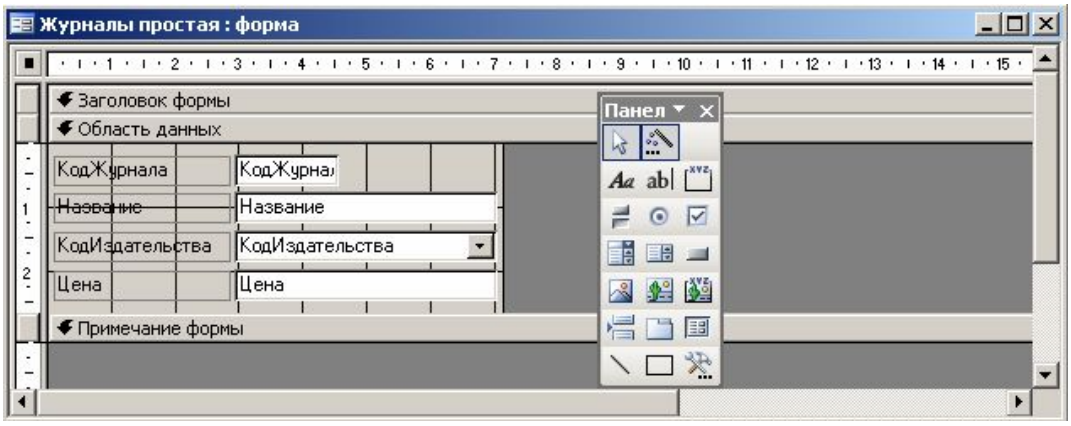
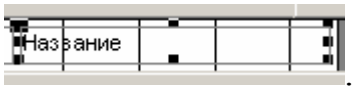



Рис. 6.9.1 . Окно формы в режиме конструктора

Если необходимо изменить размер элемента, выделите его мышью, элемент помещается в рамку с шестью маркерами выделения, левый верхний маркер чуть больше остальных. Большой маркер используется для перемещения элемента, маленькие – для изменения размера. Чтобы переместить надпись и элемент одновременно, выделите его, щелкнув по нему мышью, при этом на

границе элемента появятся маркеры 

. Установите указатель мыши над границей (но не над одним из маркеров), чтобы он принял вид руки, и перетащите в нужное место. Чтобы задать новое имя поля, щелкните на его элементе управления и отредактируйте текст.

Можно быстро выровнять несколько элементов, подогнать размеры и т.п. Для этого выделите несколько элементов, удерживая клавишу **Shift**, и в меню **Формат** выполните соответствующую команду.

Каждый элемент и сама форма имеют свойства. для отображения окна свойств выделите нужный элемент и выполните **Вид – Свойства.** Для выделения формы нажмите кнопку слева от горизонтальной линейки или выполните **Правка – Выделить форму.**

Отчет в режиме конструктора имеет вид, представленный на рис. 6.9.2. Окно состоит из зон:

- ***Заголовок отчета*** – печатается только на титульной странице, здесь можно разместить эмблему, рисунок;
- ***Верхний колонтитул*** – печатается вверху на каждой странице и обычно служит заголовками столбцов для итоговых данных;
- ***Область данных*** – представляет содержимое групп, здесь отображается и печатается каждая запись, а также вычисляемые поля;
- ***Нижний колонтитул*** – печатается внизу на каждой странице, содержит номер страницы и вычисляемые элементы управления;
- ***Примечания отчета*** – печатается после последней записи для отображения общего итога и других статистических данных (среднее значение или значение в процентах) по отчету.

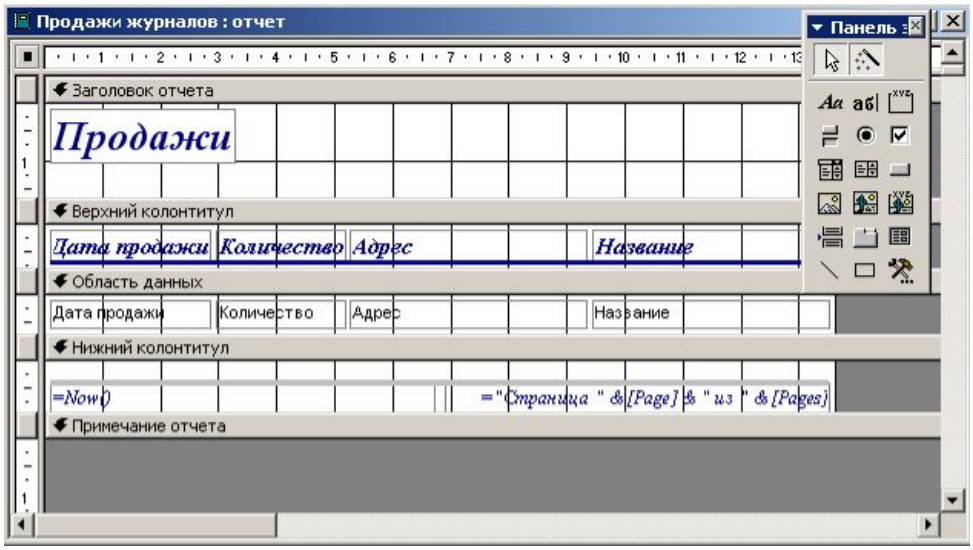



Рис. 6.9.2. Модификация отчета в режиме конструктора

В отчете в режиме конструктора можно также как и у формы, добавлять, удалять поля, изменять размер и местоположение.

# Лекція №7. АЛГОРИТМИ Й СПОСОБИ ЇХНЬОГО ОПИСУ

## 7.1. Типові структури алгоритмів

**Алгоритм** – це сувора послідовність арифметичних і логічних дій, що однозначно визначає процес обчислення результату залежно від початкових даних. Слово “алгоритм” відбулося від арабського слова “algoritmi”, що виникло з імені хорезмійського математика IX віку аль-Хорезмі.

Основними властивостями алгоритму є:

**результативність –** алгоритм повинен забезпечувати одержання результату за кінцеве число кроків;

**визначеність** – застосування алгоритму до однотипних початкових даних повинне привести до одного й тому ж результату, незалежно від користувача;

**масовість** – можливість застосування алгоритму до цілого класу однотипних задач, що розрізняються вихідними даними.

Найпоширенішими способами опису алгоритму є:

**словесно-формульний** – алгоритм записується у вигляді тексту з формулами по пунктах, що визначають послідовність дій;

**аналітичний запис** – алгоритм записується з використанням формул або спеціальних символів, наприклад, стенографічних;

**графічний у вигляді блок-схеми** – алгоритм зображується спеціальними геометричними фігурами (блоками), зв'язаними напрямними стрілками.

Зразком словесно-формульного опису алгоритму може служити наступний приклад:

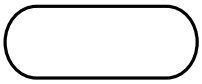
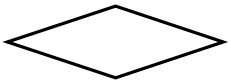
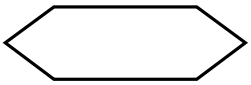
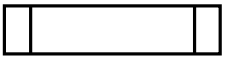
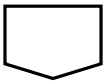
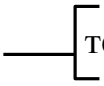
| | |
|---|---|
| обчислити значення виразу **y = 2 a - (x + 6)** | 1. *Ввести значення величин a й x.*<br>2. *Виконати додавання x і* 6 $\rightarrow$ *x*+6.<br>3. *Виконати множення a на* 2 $\rightarrow$ 2*a.*<br>4. *Відняти з отриманого добутку значення отриманої суми* $\rightarrow$ 2*a-(x*+6*).*<br>5. *Вивести значення отриманого результату y.* |

Програма, записана на якій-небудь мові програмування, також є формою подання алгоритму.

Однак найбільш зручним і наочним способом представлення алгоритму є графічний у вигляді блок-схеми. При цьому кожний логічно завершений етап обчислювального процесу зображується у вигляді спеціального геометричного символу – **блоку**. Найбільше часто використовувані графічні символи представлені в таблиці 7.1.

Таблиця 7.1

Графічні символи, які застосовуються при складанні блок-схем

| № з/п | Найменування блоку | Позначення | Виконувана дія |
|---|---|---|---|
| 1 | Початок або кінець алгоритму |  | Показує початок або кінець алгоритму |
| 2 | Ввід або вивід |  | Забезпечує ввід або вивід даних в алгоритмі |
| 3 | Арифметичний |  | Виконує арифметичні обчислення |
| 4 | Логічний |  | Виконує перевірку заданої логічної умови |
| 5 | Модифікації |  | Заголовок циклу «Для» |
| 6 | Визначений процес |  | Виклик підпрограми |
| 7 | Лінії потоку |  | Указують зв'язок і напрямок руху між блоками |
| 8 | З'єднувач |  | Указує зв'язок між перерваними лініями потоку |
| 9 | Міжсторінковий з'єднувач |  | Указує зв'язок між частинами блок-схеми, які розташовані на різних сторінках |
| 10 | Коментарі |  текст | Запис пояснення до блоку або до лінії потоку |

При складанні блок-схеми алгоритму блоки записуються послідовно один за одним і з'єднуються лініями потоку інформації, які показують напрямок руху по блок-схемі. Кожен блок алгоритму повинен мати вхід і вихід (виключення складають блоки початку й кінця алгоритму). При цьому може бути трохи вхідних у блок ліній потоку інформації й тільки один вихідний потік (виключення складають логічний блок і блок модифікації). Кілька ліній потоку можуть поєднуватися в одну лінію, але одна лінія потоку інформації не може розгалужуватися на кілька потоків. У блок-схемі будь-який шлях руху із блоку «Початок» алгоритму повинен привести в блок «Кінець» алгоритму.

У загальному випадку будь-який алгоритм може складатися із трьох частин: введення вхідних даних, обчислення необхідних величин і вивід отриманих результатів. При цьому кожний блок у блок-схемі повинен бути пронуме-

рований.

Існує три основних типових структури алгоритму:

1. Лінійний обчислювальний процес.
2. Обчислювальний процес, що розгалужується.
3. Циклічний обчислювальний процес.

Любою алгоритм складної структури може бути отриманий шляхом комбінованого використання типових структур.

## 7.2. Алгоритми лінійної структури

У **лінійному обчислювальному процесі** всі дії виконуються в суворій послідовності один за одним. Таким чином, існує тільки один шлях, по якому можна пройти із блоку «Початок» у блок «Кінець» алгоритму, тобто виконати алгоритм.

**Приклад 7.1.** Скласти алгоритм обчислення площі трикутника із заданими сторонами ***a, b*** і ***c.***

Як відомо з курсу геометрії площу трикутника, заданого довжинами сторін, можна обчислити, використовуючи формулу Герона. Розроблювальний алгоритм (рис. 7.1) повинен забезпечувати введення початкових даних, тобто значень довжин сторін трикутника ***a, b*** і ***c*** (блок 2). Потім, використовуючи введені значення, обчислювати напівпериметр ***P*** (блок 3) і значення площі трикутника ***S*** (блок 4). Після завершення обчислень необхідно вивести результат, тобто отримане значення площі трикутника ***S*** (блок 5).

Однак рішення абсолютної більшості інженерних задач неможливо звести до алгоритмів лінійної структури.

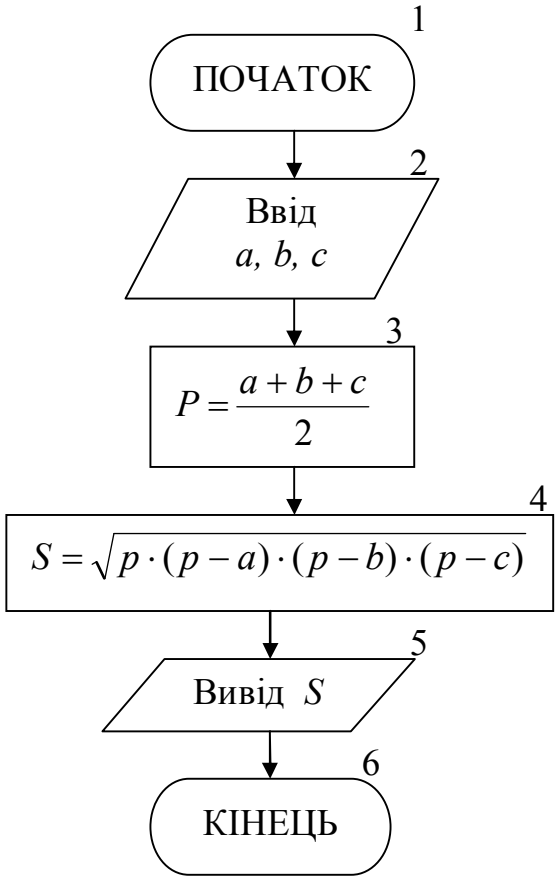



Рисунок 7.1. Алгоритм лінійної структури

## 7.3. Алгоритми структури, що розгалужується

**Обчислювальний процес, що розгалужується**, дозволяє вибрати один з декількох варіантів розв'язання поставленого завдання залежно від виконання деяких умов. Таким чином, існує кілька різних шляхів, по яких можна пройти із блоку «Початок» у блок «Кінець» алгоритму, тобто виконати алгоритм.

Для реалізації процесу вибору одного із двох варіантів розв'язання використовується логічний блок (блок перевірки умов), наведений у таблиці 7.1. При вході в цей блок (рис. 7.2) виконується перевірка логічної умови (звичайно математичної нерівності). Якщо результат перевірки умови «Істина», тобто умова виконується, то відбувається перехід до виконання блоків, що стоять по гілці «+». У противному випадку, тобто умова, що перевіряється, не виконується, відбувається перехід до виконання блоків, що стоять по гілці «–».

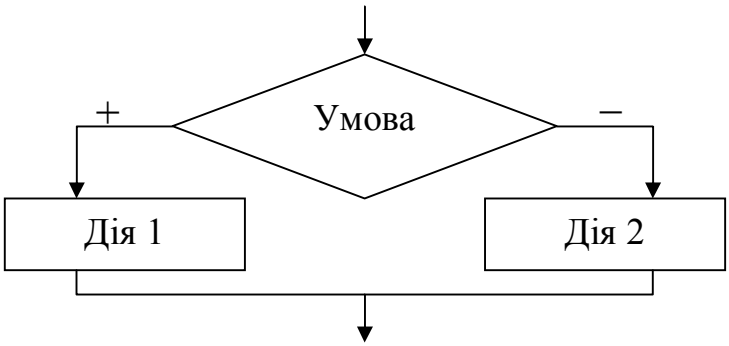



Рисунок 7.2. Використання логічного блоку

Якщо потрібно вибрати один із трьох і більш варіантів розв'язання, то необхідно використовувати вкладені логічні блоки. У випадку, коли дії повинні виконуватися тільки по одній з гілок логічного блоку, їх необхідно реалізовувати по гілці «+», підібравши відповідну умову (рис. 7.3).

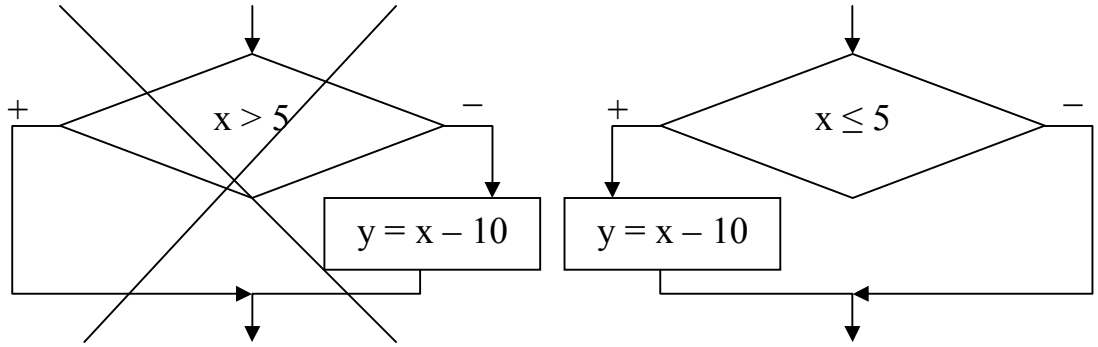



Рисунок 7.3. Раціональне використання логічного блоку

При виконанні обчислень необхідно враховувати область визначення математичних функцій. Отже, спочатку необхідно перевірити можливість обчислення даного математичного виразу при поточних значеннях початкових да-

них, тобто перевірити «**аномалію**». До «аномалій», що найбільше часто зустрічаються, належать: операція ділення (на 0 ділити не можна), обчислення квадратного кореня (підкореневий вираз повинен бути ≥ 0), обчислення логарифма (вираз під знаком логарифма повинне бути > 0), обчислення tg, ctg. У випадку виникнення «аномалії» (неможливо виконати обчислення) необхідно пропустити всі дії, які залежать від величини, що, обчислюється, і перейти в ту частину алгоритму, де можна продовжити обчислення.

**Приклад 7.2.** Скласти блок-схему алгоритму, що обчислює значення ***y*** по одній із трьох формул, залежно від значення ***x***. Блок-схема алгоритму наведена на рис. 7.4.

$$y=\begin{cases} x^2-4x, & \textit{якщо} \quad x\ge 4 \\ \sqrt{x-1.5}, & \textit{якщо} \quad 2\le x<4 \\ \dfrac{3x}{x+1}, & \textit{якщо} \quad x<2 \end{cases}$$

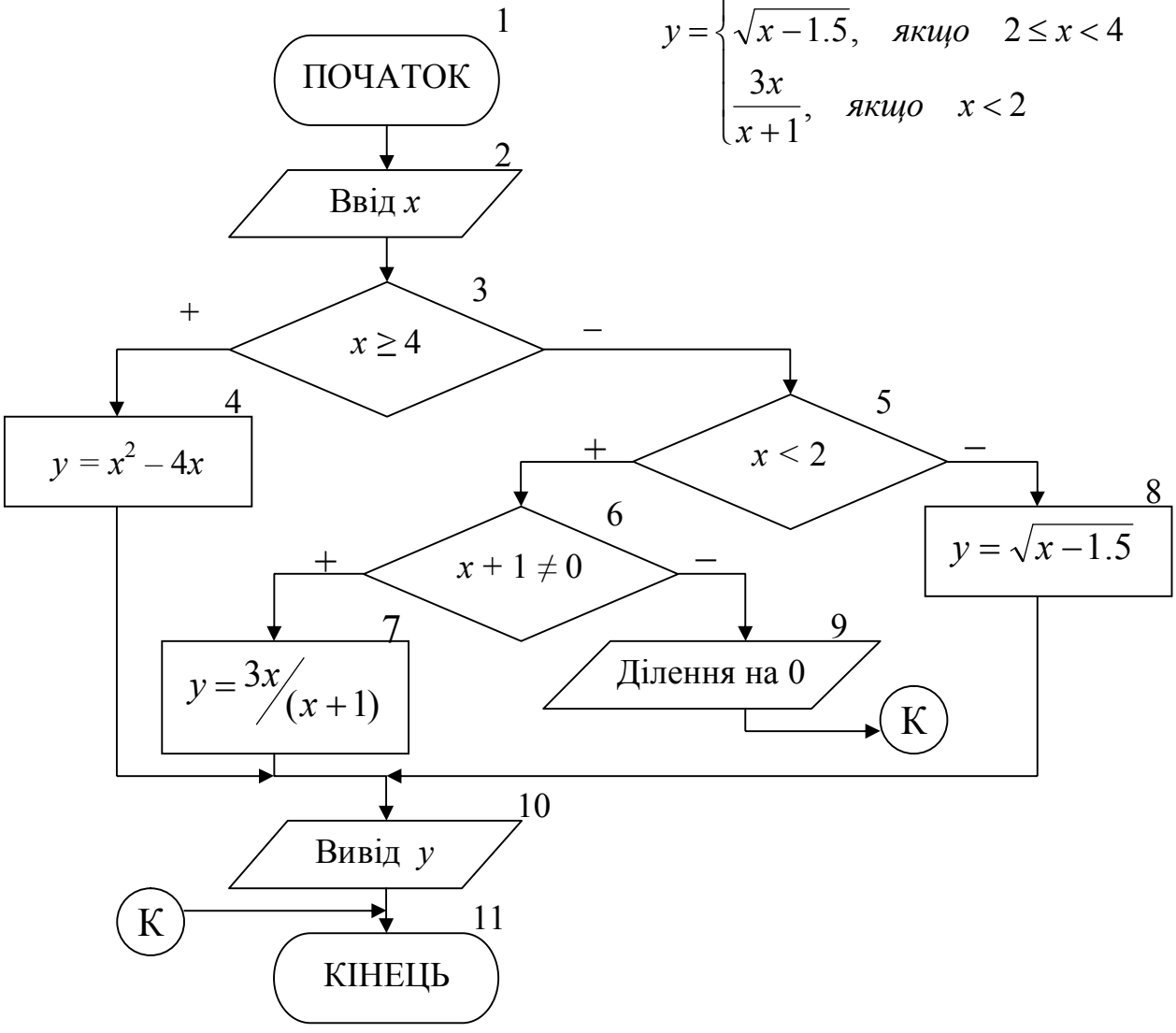



Рисунок 7.4. Алгоритм структури, що розгалужується

Для вибору формул, по яких буде обчислюватися значення ***y***, необхідно перевірити три умови. Такі обмеження є взаємовиключними умовами й досить перевірити тільки дві умови, тобто якщо не виконуються дві умови, то третя

виконується автоматично і її не треба перевіряти.

Початковими даними для розв'язання поставленої задачі є значення ***x***, що вводиться в блоці 2. Потім у блоці 3 перевіряється 1-е обмеження (***x***≥4), у випадку його виконання відбувається перехід до блоку 4, у якому обчислюється значення ***y*** по 1-й формулі. Якщо результатом перевірки 1-ї умови є «НЕПРАВДА» (це означає, що ***x*** < 4), то необхідно перевірити одне із двох умов, що залишилися. Звичайно вибирається більш «коротке» обмеження (у блоці 5 перевіряється 3-є обмеження). Якщо ***x*** < 2, то необхідно обчислювати ***y*** по 3-й формулі, у якій може виникнути «аномалія» – ділення на 0, тому в блоці 6 перевіряється нерівність знаменника дробі нулю. І тільки після цього в блоці 7 обчислюється значення ***y***. У тому випадку, коли неможливо обчислити значення ***y*** по 3-й формулі, виводиться повідомлення про виникнення помилки (блок 9) і виконується перехід на кінець алгоритму. І, нарешті, якщо значення ***x*** не задовольняє ні 1-му, ні 3-му обмеженню, виходить, виконується 2-ге обмеження й ***y*** обчислюється по 2-й формулі (блок 8). Ця формула також містить «аномалію» – підкореневий вираз повинен бути ≥ 0. Однак, перевіряти її не треба, тому що при будь-якому значенні ***x***, *що* потрапляє в інтервал [2; 4], підкореневий вираз завжди > 0. Не залежно від того, по якій формулі буде обчислене значення ***y*** його треба вивести на екран, тому виходи блоків 4, 7 і 8 поєднуються й відбувається перехід до блоку 10, у якому виводиться ***y***.

За одне виконання алгоритму може бути обчислене тільки одне значення ***y*** залежно від введеного значення ***x***. Усього ж існує 4 варіанти роботи алгоритму залежно від значення ***x***. При цьому якщо виникне «аномалія», ***y*** обчислений не буде.

## 7.4. Приклад виконання лабораторної роботи «Організація лінійного і розгалуженого обчислювальних процесів»

**Завдання**. Скласти блок-схему алгоритму, яка відповідно до вхідних даних обчислює значення заданих виразів.

Зміст звіту по лабораторній роботі.

1. Вхідні дані: *a, b*

2. Математична модель:

$$y=\begin{cases}\sqrt{a^2+1}+\sin\dfrac{\pi}{2}x, & якщо\ \ x<1.5\\ |a+x|, & якщо\ \ 1.5\le x\le 3.5\\ \sqrt{x-a}, & якщо\ \ x>3.5\end{cases}\qquad x=\begin{cases}\ln ab-1, & якщо\ \ ab>1\\ \dfrac{b-1}{a}, & якщо\ \ ab\le 1\end{cases}$$

3. Обмеження:

   а) підкореневий вираз $a^2+1 \geq 0$, **не перевіряти**, тому що $a^2+1$ завжди більше 0;

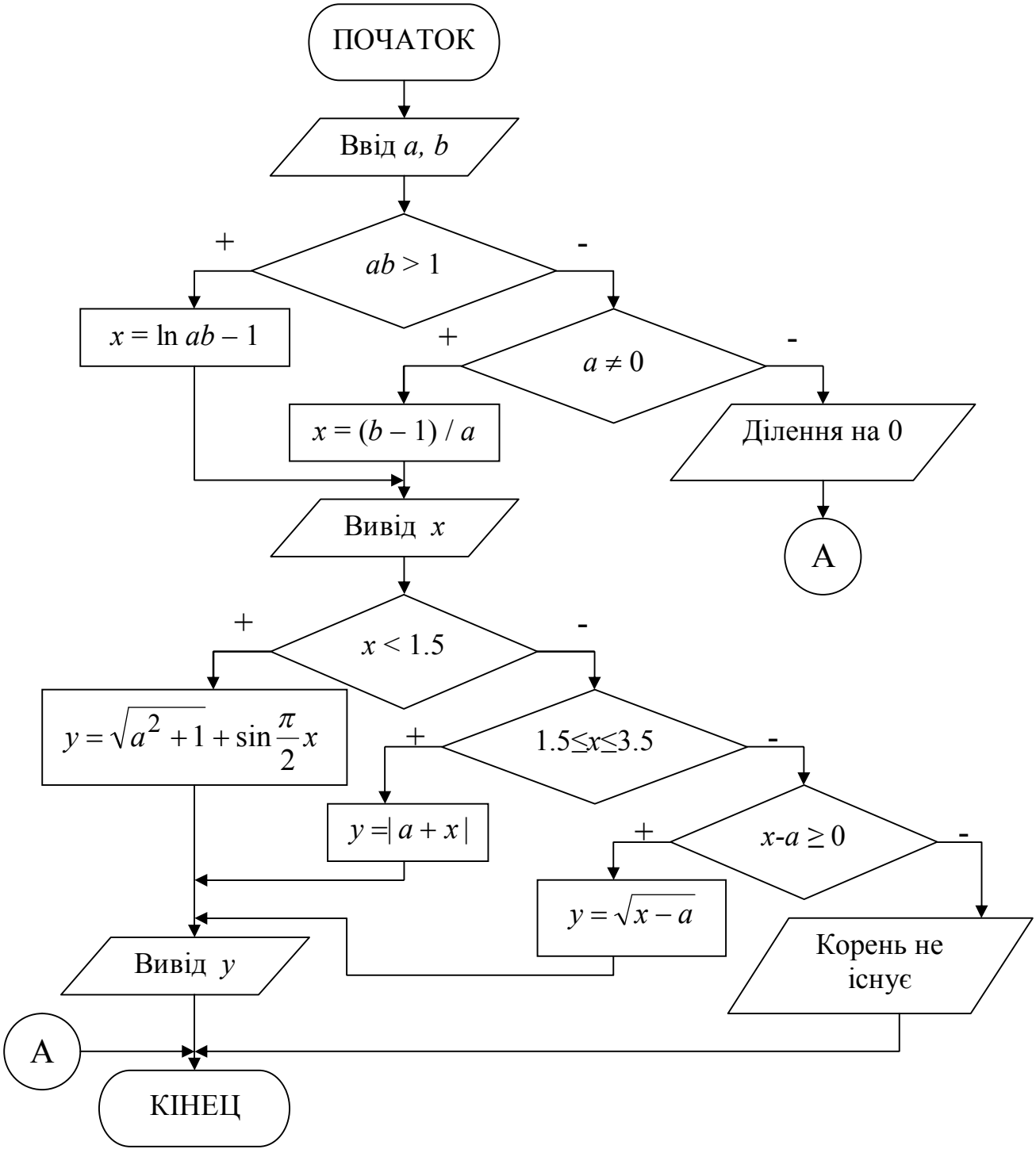
   б) підкореневий вираз $x - a \geq 0$;

   в) вираз під знаком логарифма $ab > 0$, **не перевіряти**, тому що цей вираз для обчислення $x$ використовується, тільки якщо $ab > 1$;

   г) знаменник $a \neq 0$.

4. Вихідні дані: $x, y$

5. Блок-схема алгоритму:



### 7.5. Завдання для самостійної роботи

Скласти блок-схему алгоритму рішення поставленої задачі:

1. Дано довжини трьох відрізків a, b, c. Якщо можна побудувати трикутник по цим трьох відрізках, то обчислити його периметр і площу.

2. Дано три дійсних числа a, b, c. Знайти найбільше з них.

3. Турист за день пройшов A км. До обіду він ішов t1 годин і пройшов 20 км. Ще t2 години він ішов після обіду. Коли швидкість туриста була вище: до обіду або після обіду?

4. Дано три дійсних числа a, b, c. Визначить, чи можуть ці числа бути довжинами сторін трикутника, і якщо так, те визначити вид цього трикутника (гострокутний, прямокутний або тупокутний).

5. Задано крапку M з координатами (x,y). Визначити місце розташування цієї крапки в декартової системі координат (чи є ця крапка початком координат, лежить на одній з координатних осей або розташована в одному з координатних кутів).

6. Дано три дійсних числа a, b, c. Визначить, чи можуть ці числа бути довжинами сторін трикутника, і якщо так, те визначити вид цього трикутника (рівносторонній, рівнобедрений або різнобічний).

7. Задано коло із центром у крапці $O(x_0, y_0)$ і радіусом R і крапка $A(x_1, y_1)$. Визначити місце розташування крапки стосовно кола (перебуває усередині кола, поза ним або лежить на окружності).

8. Дано рівняння двох прямих $y=a_1x+b_1$ і $y=a_2x+b_2$. Визначити, чи перетинаються ці прямі, збігаються або паралельні.

9. Задано крапки $A(a_1,a_2)$ і $B(b_1,b_2)$. Визначити, чи лежать вони на прямій y=ax+b.

10. Задано три крапки $A(a_1,a_2)$, $B(b_1,b_2)$ і $C(c_1,c_2)$. Визначити, між якими крапками відстань буде найбільшим.

## Лекція №8. ОРГАНІЗАЦІЯ АЛГОРИТМІВ ЦИКЛІЧНОЇ СТРУКТУРИ

### 8.1. Структура й основні типи циклів

В **алгоритмах циклічної структури** виконання тих самих дій може повторюватися кілька разів. Організація циклічного обчислювального процесу виконується в кілька етапів:

1-й етап – **підготовка до виконання циклу**. На цьому етапі задаються початкові значення для параметра циклу й змінних, що використовуються для зберігання величин, що накопичуються (сума, кількість або добуток величин). **Параметр циклу** – це змінна, на основі якої будується цикл. Вона повинна за-

довольняти трьом умовам: бути вхідною величиною для виконання обчислень; змінюватися за певним законом (найчастіше це закон арифметичної прогресії); впливати на умову завершення повторюваних обчислень.

2-й етап – **тіло циклу**. Воно містить арифметичні й логічні дії, які можуть повторюватися певну кількість разів. Наприкінці тіла циклу обов'язково повинен бути блок, у якому змінюється значення параметра циклу.

3-й етап – **умова виходу із циклу**, що запобігає нескінченне виконання циклу. За допомогою цієї умови перевіряється чи треба повторювати обчислення, або виходити із циклу.

Блок-схема, яка показує етапи організації циклу представлена на рис. 8.1. Така організація циклічного обчислювального процесу називається **циклом з пістумовою**. У цьому випадку, після кожного виконання тіла циклу, перевіряється умова виходу із циклу. Якщо вона не виконується, то відбувається повернення на початок тіла циклу й повторення обчислень. Коли умова виходу буде виконано, відбудеться завершення роботи й вихід із циклу. Наявність лінії повернення в блок-схемі є основною ознакою циклічного обчислювального процесу.

Крім циклу з **пістумовою** також існують цикл із **передумовою** і цикл «Для».

У **циклі із передумовою** (рис. 8.2) перед початком тіла циклу перевіряється **умова продовження циклу**. При цьому якщо умова продовження циклу є істиною, то виконуються дії, що становлять тіло циклу, і відбувається перехід у початок на перевірку умови. Цикл завершує свою роботу в тому випадку якщо умова продовження циклу не виконується – стає неправдою. Таким чином, у циклі з **пістумовою** на відміну від циклу із **передумовою**, тіло циклу завжди виконається хоча б один раз.


Рисунок 8.1. Організація циклу із пістумовою

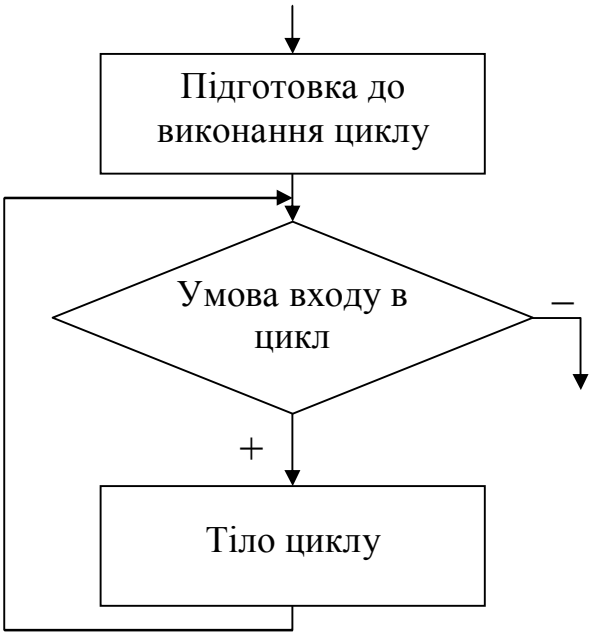



Рисунок 8.2. Організація циклу із передумовою

**Цикл «Для»** реалізується на основі блоку модифікації і являє собою варіант циклу із **передумовою**, у якому передбачається, що частина дій по організації циклу виконується автоматично. Робота циклу «Для» більш докладно буде розглянута в наступних лекціях.

Цикл, до складу якого не входять інші цикли, називається простим.

**Приклад 8.1.** Скласти блок-схему алгоритму, що обчислює значення функції $y = 2 \cdot x + 0.5$, при різних значеннях $x$, що належать інтервалу

$1 \le x \le 3$, $hx = 0.5$.

Для рішення поставленої задачі необхідно організувати перебір всіх можливих значень $x$ із заданого інтервалу із зазначеним кроком (1; 1.5; 2; 2.5; 3). Для кожного отриманого $x$ обчислити значення функції $y$, тобто організувати циклічний обчислювальний процес. Блок-схема алгоритму, у якій використовується цикл із пістумовою, наведена на рисунку 8.3.

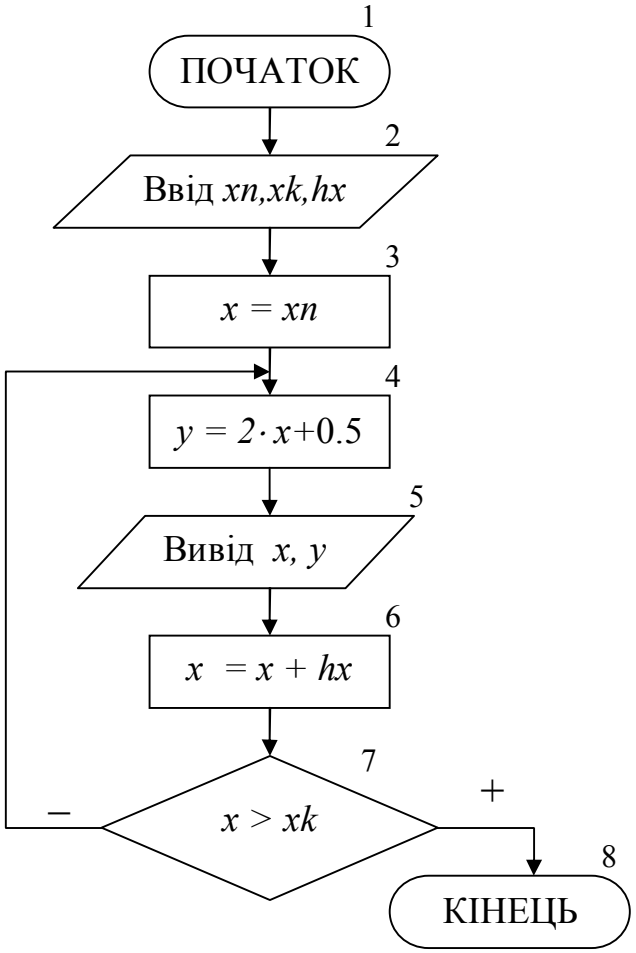



Рисунок 8.3. Блок-схема для прикладу 8.1

Початковими даними для рішення поставленої задачі є ***xn, xk, hx*** (початкового, кінцеве значення $x$ із заданого інтервалу й крок зміни ***x)***, які вводяться в блоці 2. У нашому випадку необхідно буде ввести $xn = 1$, $xk = 3$, $hx = 0.5$.

Така організація введення початкових даних забезпечує відповідність

блок-схеми двом властивостям алгоритму – визначеність і масовість, і надалі буде називатися **складанням алгоритму в «загальному виді»** (всі початкові дані повинні вводитися, а не присвоюватися в процесі виконання алгоритму).

Потім відбувається підготовка до виконання циклу – блок 3, у якому параметру циклу присвоюється початкове значення. Як параметр циклу обраний змінна ***x***, що задовольняє трьом умовам (***x*** є вхідним значенням для обчислень, змінюється за законом арифметичної прогресії, і впливає на завершення роботи циклу). Після цього починається тіло циклу (блоки 4-7). Обчислюється значення ***y*** (блок 4). Розраховане значення ***y*** виводиться на екран одночасно з відповідним значенням параметра циклу ***x*** (блок 5). Відбувається зміна параметра циклу (блок 6), таким чином, утворюється наступне значення ***x*** із заданого інтервалу. Наприкінці тіла циклу стоїть умова виходу із циклу ***x > xk*** (блок 7), за допомогою якого перевіряється, чи не вийшло ***нове*** значення ***x*** за праву границю заданого інтервалу. Якщо значення ***x*** не перевищує ***xk***, то відбувається повернення на початок тіла циклу (блок 4) і повторюються ті ж самі дії. У випадку виконання умови відбувається вихід із циклу (перехід у блок 8) і алгоритм завершує свою роботу.

Однократне виконання тіла циклу називається **кроком**. За один крок обчислюється одне значення результуючої функції при одному (поточному) значенні параметра циклу. Результати покрокового виконання циклу, наведеного на рис. 8.3, представлені в таблиці 8.1.

Кількість кроків виконання циклу ***N*** можна обчислити до початку роботи алгоритму по наступній формулі (8.1):

$$N = \left]\frac{xk - xn}{hx}\right[ + 1 \qquad (8.1)$$

де ] [ позначають цілу частину виразу.

Таблиця 8.1. Покрокове виконання циклу

| № кроку | Поточне значення ***x*** (на початку кроку) | Обчислене значення ***y*** | Нове значення ***x*** (наприкінці кроку) | Результат перевірки умови виходу із циклу (***x > xk***) |
|---|---|---|---|---|
| 1 | ***1*** (***xn)*** | 2,5 | 1,5 | Неправда |
| 2 | 1,5 | 3,5 | 2 | Неправда |
| 3 | 2 | 4,5 | 2,5 | Неправда |
| 4 | 2,5 | 5,5 | 3 | Неправда |
| 5 | ***3*** (***xk***) | 6,5 | ***3,5*** | Істина |

Циклічні обчислювальні процеси, для яких можна обчислити кількість

кроків циклу без виконання алгоритму, називаються **циклами з відомим числом повторень**. Для реалізації циклів з відомим числом повторень можна рівноцінно використовувати кожний із трьох стандартних типів циклу (з пістумовою, із передумовою, «Для»).

Якщо в циклі при виконанні обчислень може виникнути «**аномалія**», то завершувати виконання алгоритму, як в обчислювальному процесі, що розгалужується, не треба. У цьому випадку необхідно пропустити всі операції, що використовують змінну, значення якої неможливо обчислити, і виконати перехід до блоку, у якому змінюється значення параметра циклу, тобто продовжити роботу циклу. При наступному значенні параметра циклу «аномалія» може не виникнути, і робота циклу піде природним шляхом.

## 8.2. Алгоритми знаходження суми, добутку й кількості обчислених значень

Часто поряд із циклічним обчисленням значень величини або сукупності величин необхідно визначити суму, добуток або кількість всіх одержуваних значень або деяких з них.

При виконанні підсумовування й обчисленні добутку або кількості раціонально використовувати принцип поступового нагромадження значень величин, які одержувані на кожному кроці циклу. Для зберігання величин, що накопичуються, використовуються додаткові змінні, котрим попередньо необхідно задати початкові значення. Звичайно це робиться на етапі підготовки до виконання циклу (див. п. 8.1). **Як початкове значення для суми й кількості використовується нуль, для добутку - одиниця.**

Для пояснення принципу обчислення величин, що накопичуються, розглянемо приклад аналогічний прикладу з п. 8.1.

**Приклад 8.2.** Скласти блок-схему алгоритму, що обчислює значення функції $y = 2 \cdot x + 1$, при різних значеннях параметра $x$, що належать інтервалу $-3 \le x \le 3$, $hx = 1$. Як додаткова задача необхідно знайти: кількість ($k$) обчислених значень $y$, які $\le 0$; суму ($S$) всіх значень $y$ ($S = \Sigma y$) і добуток ($P$) значень $y$, які $> 0$ ($P = \prod_{y>0} y$ ).

При рішенні цієї задачі необхідно організувати перебір всіх можливих значень $x$ із заданого інтервалу із зазначеним кроком, і для кожного отриманого $x$ обчислити значення функції $y$. Розроблювальний алгоритм є циклом з відомим числом повторень, тому що кількість повторень обчислень можна визначити по формулі (8.1):

$$N = \left]\frac{3-(-3)}{1}\right[ + 1 = 7$$

Тому для реалізації алгоритму, на відміну від прикладу 8.1, можна скористатися циклом із передумовою, блок-схема якого наведена на рис. 8.4.

Початковими даними, як і в прикладі 8.1, є ***xn, xk, hx***, які вводяться в блоці 2. У нашому випадку необхідно буде ввести ***xn*** =-3**,** ***xk*** =3**,** ***hx*** =1. На етапі підготовки до виконання циклу задаються початкові значення для параметра циклу ***x*** (блок 3) і додаткових величин ***k, S, P*** (блок 4). Потім починається тіло циклу із передумовою (блоки 5-12).

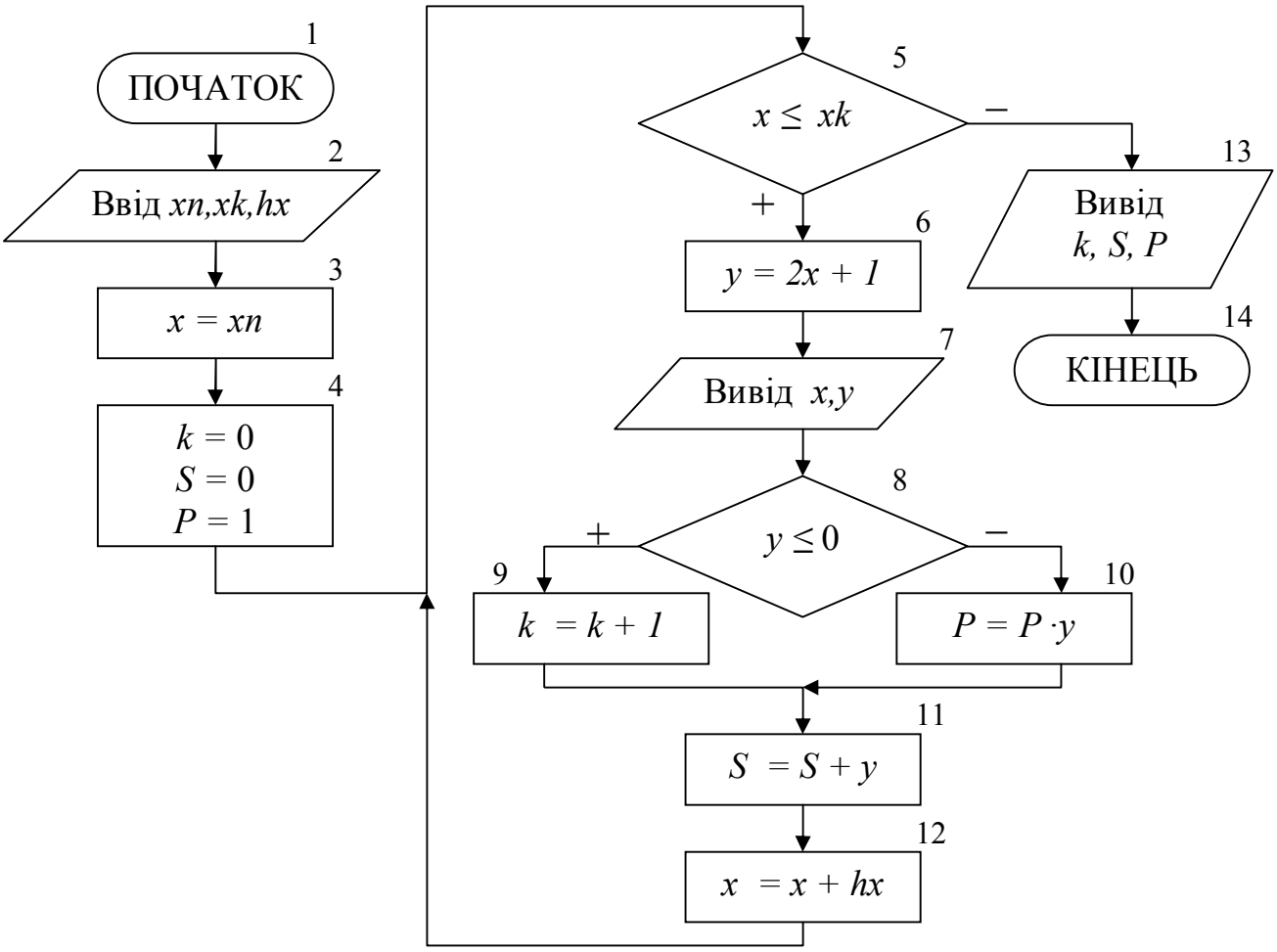



Рисунок 8.4. Блок-схема для прикладу 8.2

При входе в цикл проверяется условие продолжения вычислений в теле цикла (блок 5). Вычисляется (блок 6) и выводится на экран (блок 7) значение ***y*** при текущем значении параметра цикла ***x***. Затем проверяется вычисленное значение ***y*** (блок 8), и если ***y*** ≤ 0, то ***k*** увеличивается на единицу (блок 9), в противном случае текущее значение ***y*** учитывается в произведении ***P*** (блок 10). Кроме этого, каждое вычисленное значение ***y*** добавляется к сумме ***S*** (блок 11). В конце тела цикла выполняется изменение параметра цикла (блок 12), т. е. вычисляется следующее значение ***x*** из заданного интервала, и происходит возврат в начало цикла. После выхода из цикла выводятся значения переменных ***k, P, S*** (блок 13).

Результати покрокового виконання циклу в алгоритмі, який наведено на рис. 8.4, представлені в таблиці 8.2.

Покрокове виконання циклу показує, що підсумкові значення суми, кількості й добутку (***k=3, P=105, S=7***) будуть отримані після завершення роботи циклу. Тому вони повинні виводитися **тільки після виходу із циклу**. Якщо вивід ***k, P, S*** організувати усередині тіла циклу, то будуть виведені всі проміжні значення цих величин, які наведені в таблиці. Також буде неправильно поставити вивід значення параметра циклу ***x*** і величини, що обчислюється ***y*** (блок 7), після виходу із циклу. З таблиці 8.2 видно, що в цьому випадку будуть виведені останні значення, що зберігаються в цих змінних (***x***=4 і ***y***=7).

Таблиця 8.2. Покрокове виконання циклу

| № кроку | Поточне значення ***x*** (на початку кроку) | Результат перевірки умови продовження циклу (***x*** ≤ ***xk***) | Обчислене значення ***y*** | Поточні значення величин, що накопичуються ***k, P, S*** | Нове значення ***x*** (наприкінці кроку) |
|---|---|---|---|---|---|
| 1 | ***-3*** (***xn)*** | Істина | -5 | ***k***=1, ***P***=1, ***S***=-5 | -2 |
| 2 | -2 | Істина | -3 | ***k***=2, ***P***=1, ***S***=-8 | -1 |
| 3 | -1 | Істина | -1 | ***k***=3, ***P***=1, ***S***=-9 | 0 |
| 4 | 0 | Істина | 1 | ***k***=3, ***P***=1, ***S***=-8 | 1 |
| 5 | 1 | Істина | 3 | ***k***=3, ***P***=3, ***S***=-5 | 2 |
| 6 | 2 | Істина | 5 | ***k***=3, ***P***=15, ***S***=0 | 3 |
| 7 | ***3*** (***xk***) | Істина | 7 | ***k***=3, ***P***=105, ***S***=7 | 4 |
| 8 | ***4*** | Неправда | | | |

## 8.3. Приклад виконання лабораторної роботи «Організація циклів з відомим числом повторень»

**Завдання**. Скласти блок-схему алгоритму для обчислення значень *y* при всіх можливих значеннях *x*, які лежать в інтервалі від *xn* до *xk* із кроком *hx*. Використовувати цикл із пістумовою.

Зміст звіту по лабораторній роботі.

1. Вхідні дані: *a, xn, xk, hx*

2. Математична модель:

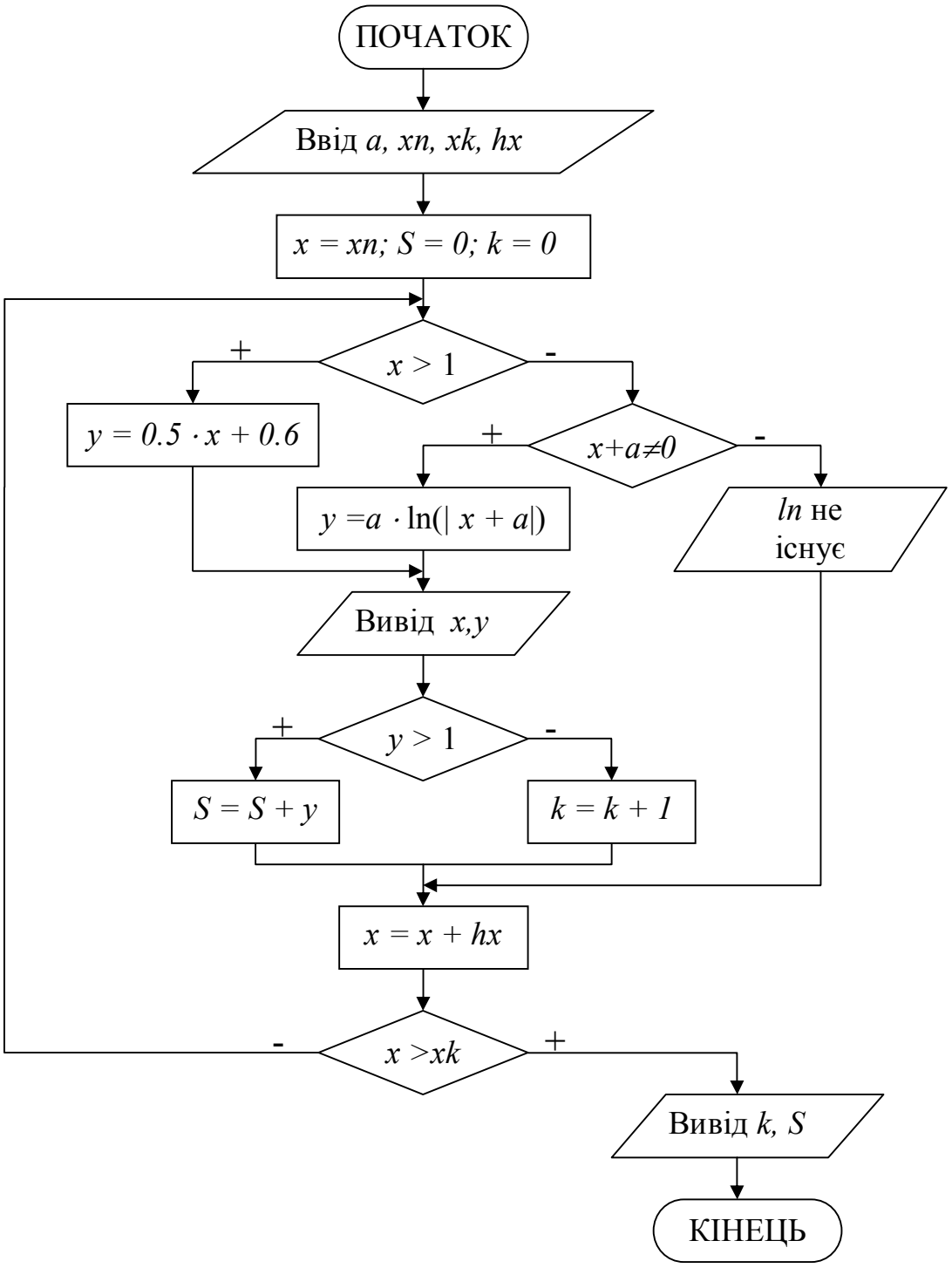
$$y = \begin{cases} 0.5x + 0.6, & якщо \quad x > 1 \\ a \cdot \ln(|x + a|), & якщо \quad x \le 1 \end{cases}$$

Обчислити $S$ – суму значень $y > 1$ і $k$ – кількість $y \le 1$.

3. Обмеження: вираз під знаком логарифма $x + a \neq 0$

4. Вихідні дані: *x, y, S* і *k*.

5. Блок-схема алгоритму:



### 8.4. Завдання для самостійної роботи

Скласти блок-схему алгоритму, що при різних значеннях параметра ***x***, що належать інтервалу ***xn*** ≤ ***x*** ≤ ***xk***, із кроком ***hx***, обчислює:

1. Середнє арифметичне додатних значень функції $y = x/2 - 1$.

2. $K = N!$, де $N$ – кількість парних значень функції $y = x^2 - 1$ (***xn, xk***, ***hx*** – цілі числа).

3. $P = \prod_{y<10} y + \sum_{y>10} y$ значень функції $y = x^2 - x/2$.

4. Кількість дійсних корінь рівняння $ay^2 + xy + b = 0$ (*a, b* – ввести).

5. Різницю між середнім арифметичним від'ємних значень і середнього геометричним додатних значень функції $y = x^{2/2} - x^{3/3}$.

6. Відсоток позитивних, негативних і нульових значень функції $y = sin(x^2) - 0{,}2$.

7. Середнє арифметичне непарних значень функції $y = 2 \cdot x - 1$ (***xn, xk***, ***hx*** – цілі числа).

8. Значення функції $y = cos(x^2)$ і визначає, чи утворять значення ***y*** знакочергуєму послідовність.

9. Значення функції $y = cos(x+2)$ і, уважаючи (***x, y***) – координатами точок на площині, визначає кількість точок розташованих в 1 і 3 квадрантах.

10. Середнє арифметичне перших п'яти додатних значень функції $y = sin(x)^2 - 0.3$.

## Лекція №9. АЛГОРИТМИ ЦИКЛІЧНОЇ СТРУКТУРИ

### 9.1. Цикли з невідомим числом повторень

При рішенні деяких задач може виникнути ситуація, коли неможливо заздалегідь визначити кількість повторень обчислень. У таких випадках мова йде про **цикли з невідомим числом повторень**.

У циклах з невідомим числом повторень обчислювальний процес завершується при виконанні деякої додаткової умови. Значення параметра циклу вже не задається у вигляді діапазону, а тільки вказується його початкове значення й крок зміни. Проте, організація циклу виконується за стандартною методикою, указаної в п. 8.1. Відмінність полягає в тім, що не будь-який тип циклічного обчислювального процесу можна використати. Тип циклу визначається відповідно до заданої додаткової умови завершення обчислень. Це однозначно виключає можливість використання циклу «Для» на основі блоку модифікації. Для визначення кількості кроків повторення циклу необхідно організувати лічильник.

**Приклад 9.1.** Скласти блок-схему алгоритму, що обчислює значення функції $y = ln(1 + 0.2 \cdot x)$, при різних значеннях параметра ***x***, що змінюється від ***x?-***

$n \le 3$ із кроком $hx = -2$. При цьому обчислювати $y$ доти, поки вираз під знаком логарифма залишається більше 0. Визначити кількість ($k$) обчислених значень $y$.

Перед рішенням задачі необхідно визначити тип циклу, що буде використовуватися. Вираз, що обчислюється, містить «аномалію » (значення під знаком логарифма повинне бути більше 0) , що у теж час є умовою завершення обчислень. Інакше кажучи, необхідно спочатку перевірити можливість обчислення значення $y$, а тільки потім обчислювати його. Тому в цьому випадку для організації обчислювального процесу можна використати тільки цикл із передумовою (рис. 9.1).

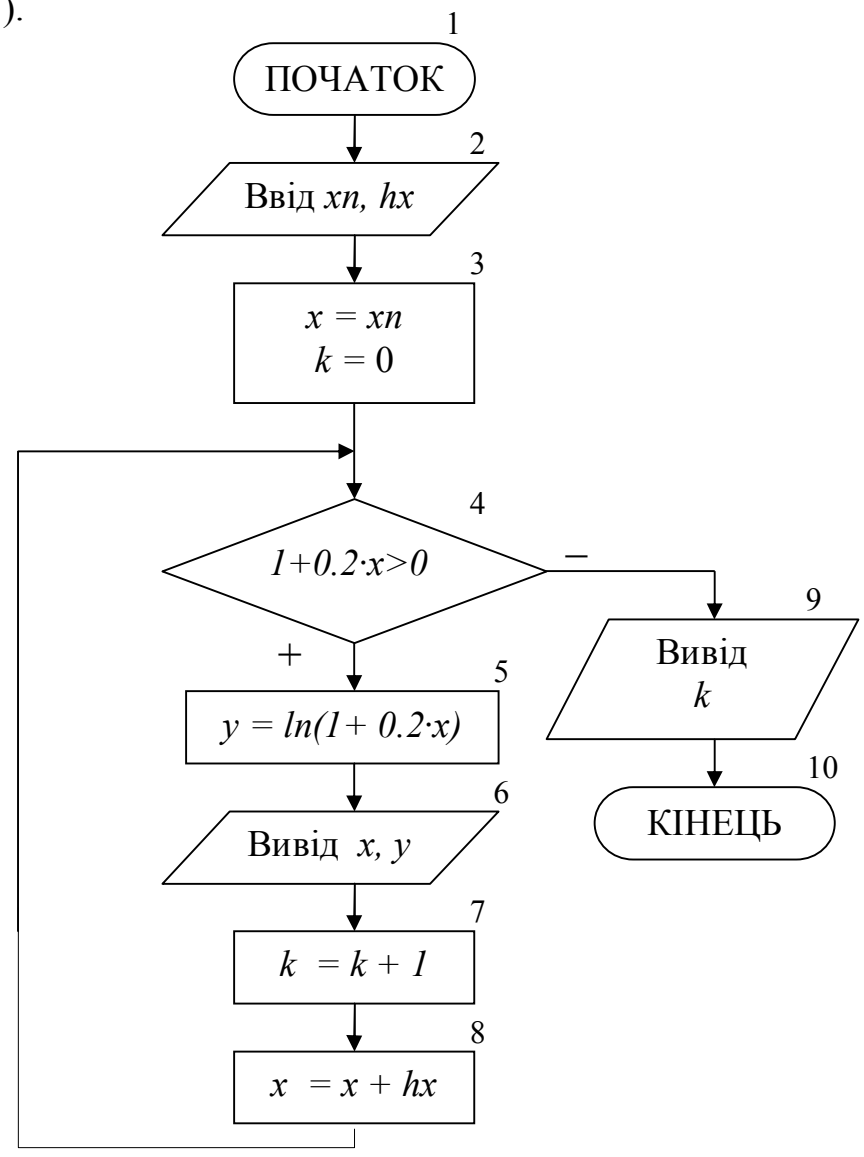



Рисунок 9.1. Блок-схема для прикладу 9.1

У якості початкових даних вводяться значення змінних ***xn, hx*** (блок 2). У блоці 3 задаються початкові значення для параметра циклу ***x*** і лічильника кількості повторень циклу ***k***. Після цього, у блоці 4 одночасно перевіряються умова продовження циклу й можлива «аномалія» (вираз під знаком логарифма повинен бути більше 0). У тілі циклу для поточного значення ***x*** обчислюється й ви-

водиться відповідне значення ***y*** (блоки 5-6), а також збільшується значення лічильника обчислених ***y*** (блок 7). Наприкінці тіла циклу виконується перехід до наступного значення параметра циклу ***x*** (блок 8). Цикл працює доти, поки під знаком логарифма не з'явиться вираз ≤ 0. Після виходу із циклу в блоці 9 виводиться змінна ***k*** – кількість обчислених значень ***y***.

Результати покрокового виконання циклу в алгоритмі, наведеному на рис. 9.1, представлені в таблиці 9.1.

Таблиця 9.1. Покрокове виконання циклу

| № кроку | Поточне значення ***x*** (на початку кроку) | Результат перевірки умови продовження циклу (***1+0.2·x>0***) | Обчислене значення ***y*** | Поточне значення ***k*** | Нове значення ***x*** (наприкінці кроку) |
|---|---|---|---|---|---|
| 1 | ***3 (xn)*** | Істина (*1+0.2·3=1.6 >0*) | 0.47 | ***k***=1 | 1 |
| 2 | 1 | Істина (*1+0.2·1=1.2 >0*) | 0.18 | ***k***=2 | -1 |
| 3 | -1 | Істина (*1+0.2-1=0.8 >0*) | -0.22 | ***k***=3 | -3 |
| 4 | -3 | Істина (*1+0.2-3=0.4 >0*) | -0.92 | ***k***=4 | -5 |
| 5 | ***-5*** | Неправда (*1+0.2-5=0 >0*) | | | |

## 9.2. Вкладені цикли

Рішення деяких задач вимагає виконання перебору значень не однієї, а декількох величин одночасно. У цьому випадку мова йде про застосування **вкладених циклів**, кожний з яких організовується по стандартному принципу (може бути кожного із трьох типів) і здійснює перебір тільки одного параметра. При цьому перший цикл називається зовнішнім, а вкладені в нього – внутрішніми. Причому той самий цикл може бути зовнішнім стосовно одному й внутрішнім стосовно іншого циклу. Границі внутрішнього циклу не можуть виходити за межі зовнішнього стосовно нього циклу.

Для кожного значення параметра зовнішнього циклу відбувається перебір всіх можливих значень параметра внутрішнього циклу. Інакше кажучи, завжди виконується в першу чергу самий внутрішній цикл. Така організація циклів дає можливість перебрати значення їхніх параметрів у всіх можливих комбінаціях.

**Приклад 9.2.** Скласти блок-схему алгоритму, що обчислює значення функції ***y = a - 2· b***, для всіх можливих комбінацій значень параметрів ***a*** і ***b***, що належать інтервалам $0 \le a \le 2$, $ha = 1$ і $-2 \le b \le 2$, $hb = 2$, відповідно. Визначити середнє арифметичне (***S***) обчислених значень ***y***.

Для рішення поставленої задачі необхідно організувати два вкладених

цикли по перебору параметрів ***a*** і ***b***. При цьому кожний цикл є циклом з відомим числом повторень, тому що заздалегідь можна обчислити кількість значень ***a*** із заданого інтервалу (***Na*** **= ](2 – 0)/1[ + 1 = 3**) і кількість значень ***b*** із заданого інтервалу (***Nb*** **= ](2 – (-2))/2[ + 1 = 3**), а отже й загальну кількість обчислених значень ***y*** (***N*** **=** ***Na*** **·** ***Nb***). Виходить, для організації циклів можна використовувати кожний із трьох стандартних типів.

Крім цього ***y*** одночасно залежить від двох параметрів ***a*** і ***b***, тому не має принципового значення по якому параметру робити зовнішній цикл, а по якому – внутрішній (зовнішнім може бути цикл по параметру ***a***, а внутрішнім – по параметру ***b***, і навпаки). У випадку ж якщо одна з величин, що обчислюють, залежить тільки від одного параметра, то цикл по цьому параметру раціонально зробити зовнішнім і обчислювати величини, що залежать тільки від цього параметра в зовнішньому циклі. Така організація дозволить уникнути багаторазового обчислення тих самих значень у внутрішньому циклі.

На рисунку 9.2 представлена блок-схема алгоритму, у якій для рішення поставленої задачі, використовується два вкладених цикли: зовнішній цикл із передумовою по параметру ***a*** і внутрішній цикл із пістумовою по параметру ***b***.

Робота алгоритму починається з введення вхідних даних (блок 2), які являють собою початкові, кінцеві значення й крок зміни параметрів ***a*** і ***b*** із заданих інтервалів. На етапі підготовки до виконання зовнішнього циклу (блок 3) параметру циклу ***a*** присвоюється початкове значення ***an***, а також обнульовуються змінні ***S*** і ***k***, які використовувані для зберігання суми й кількості обчислених значень ***y***.

Тіло зовнішнього циклу із передумовою включає блоки 4-11. Якщо поточне значення ***a*** не виходить за праву границю інтервалу (блок 4), то відбувається виконання тіла зовнішнього циклу, що містить у собі внутрішній цикл. У блоці 5 відбувається підготовка до виконання внутрішнього циклу, його параметру ***b*** присвоюється початкове значення ***bn***.

Тіло внутрішнього циклу з пістумовою включає блоки 6-10. У блоці 6 обчислюється поточне значення ***y***, а в блоці 7 відбувається вивід обчисленого ***y*** і відповідних йому значень ***a*** і ***b***. У блоці 8 отримане значення ***y*** додається до суми ***S*** і збільшується значення лічильника обчислених ***y***. Потім відбувається зміна параметра внутрішнього циклу ***b*** на значення кроку ***hb*** (блок 9) і виконується перевірка умови виходу із внутрішнього циклу (блок 10).

Робота алгоритму починається з введення вхідних даних (блок 2), які являють собою початкові, кінцеві значення й крок зміни параметрів ***a*** і ***b*** із заданих інтервалів. На етапі підготовки до виконання зовнішнього циклу (блок 3) параметру циклу ***a*** присвоюється початкове значення ***an***, а також обнульовуються змінні ***S*** і ***k***, які використовувані для зберігання суми й кількості обчислених значень ***y***.

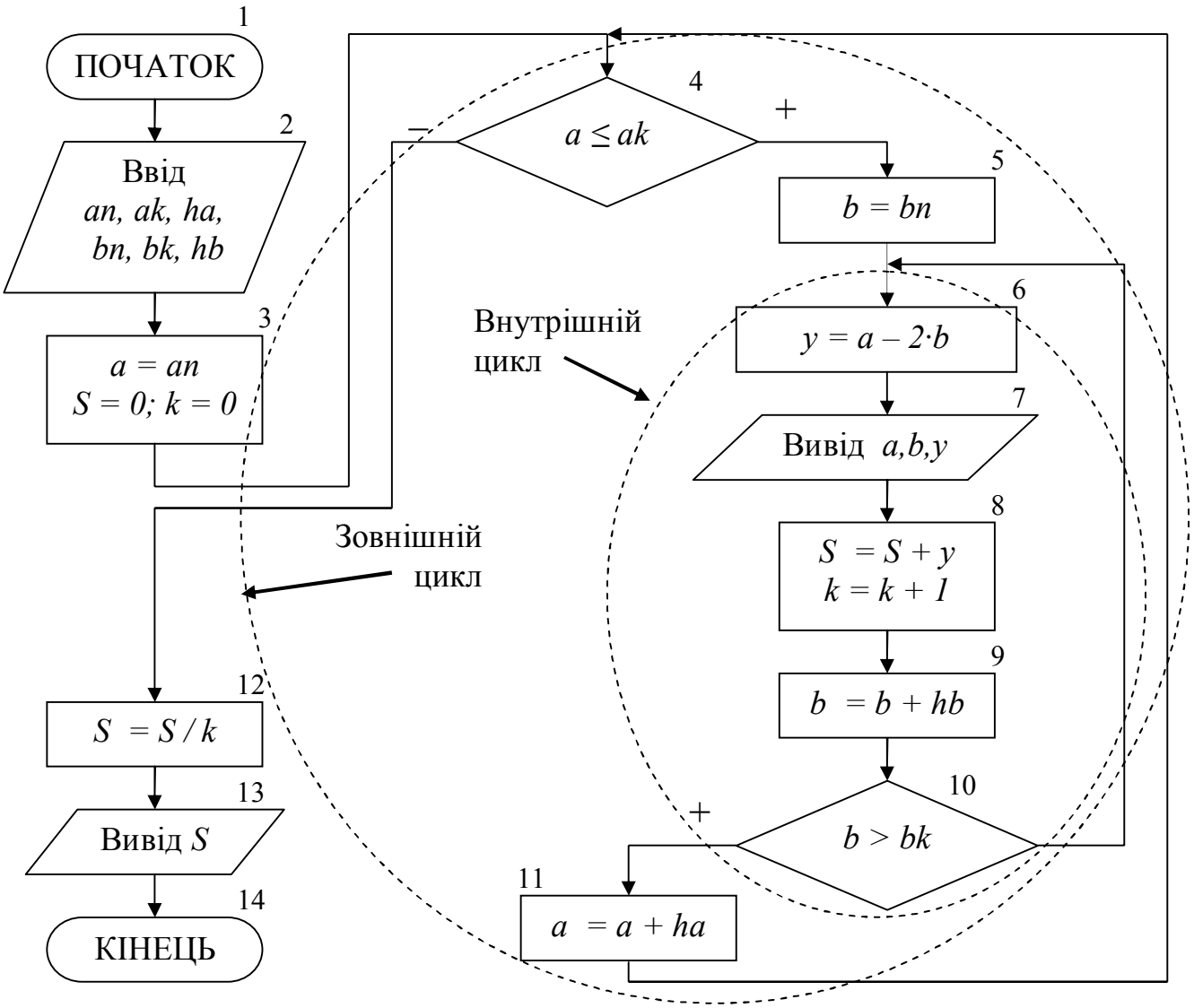



Рисунок 9.2. Блок-схема для прикладу 9.2

Тіло зовнішнього циклу із передумовою включає блоки 4-11. Якщо поточне значення ***a*** не виходить за праву границю інтервалу (блок 4), то відбувається виконання тіла зовнішнього циклу, що містить у собі внутрішній цикл. У блоці 5 відбувається підготовка до виконання внутрішнього циклу, його параметру ***b*** присвоюється початкове значення ***bn***.

Тіло внутрішнього циклу з пістумовою включає блоки 6-10. У блоці 6 обчислюється поточне значення ***y***, а в блоці 7 відбувається вивід обчисленого ***y*** і відповідних йому значень ***a*** і ***b***. У блоці 8 отримане значення ***y*** додається до суми ***S*** і збільшується значення лічильника обчислених ***y***. Потім відбувається зміна параметра внутрішнього циклу ***b*** на значення кроку ***hb*** (блок 9) і виконується перевірка умови виходу із внутрішнього циклу (блок 10).

Якщо поточне значення ***b*** не перевищує кінцевого ***bk***, то відбувається повернення на початок внутрішнього циклу й повторюється обчислення ***y*** при новому значенні ***b***. У теж час значення параметра зовнішнього циклу ***a*** залишається незмінним. Таким чином, при одному значенні параметра зовнішнього циклу відбувається перебір всіх значень параметра внутрішнього циклу.

Після виходу із внутрішнього циклу в блоці 11 відбувається зміна параметра зовнішнього циклу ***a*** на значення кроку ***ha***. Потім виконується перехід на початок циклу (блок 4) і повторюються описані вище дії, поки не завершить роботу зовнішній цикл. Після виходу із зовнішнього циклу в блоках 12-13 обчислюється й виводиться ***S*** – середнє арифметичне обчислених значень ***y***.

Результати покрокового виконання циклів в алгоритмі, приведеному на рис. 9.2, представлені в таблиці 9.2.

Таблиця 9.2. Покрокове виконання алгоритму з вкладеними циклами

| № кроку | Поточне значення ***a*** (на початку кроку) | Перевірка умови продовження зовнішнього циклу (***a ≤ ak***) | Поточне значення ***b*** (на початку кроку) | Обчислене значення ***y*** | Поточні значення ***k, S*** | Нове значення ***b*** (наприкінці кроку) | Перевірка умови виходу із внутрішнього циклу (***b > bk***) | Нове значення ***a*** (наприкінці кроку) |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 (***an)*** | Істина | -2 (***bn)*** | 4 | ***k***=1, ***S***=4 | 0 | Неправда | - |
| 2 | - | - | 0 | 0 | ***k***=2, ***S***=4 | 2 | Неправда | - |
| 3 | - | - | 2 (***bk)*** | -4 | ***k***=3, ***S***=0 | 4 | Істина | 1 |
| 4 | 1 | Істина | -2 (***bn)*** | 5 | ***k***=4, ***S***=5 | 0 | Неправда | - |
| 5 | - | - | 0 | 1 | ***k***=5, ***S***=6 | 2 | Неправда | - |
| 6 | - | - | 2(***bk)*** | -3 | ***k***=6, ***S***=3 | 4 | Істина | 2 |
| 7 | 2 (***ak)*** | Істина | -2 (***bn)*** | 6 | ***k***=7, ***S***=9 | 0 | Неправда | - |
| 8 | - | - | 0 | 2 | ***k***=8, ***S***=11 | 2 | Неправда | - |
| 9 | - | - | 2(***bk)*** | -2 | ***k***=9, ***S***=9 | 4 | Істина | 3 |
| 10 | 3 | Неправда | - | - | - | - | - | - |

Варто також звернути увагу на те, що початкове значення параметра вну-

трішнього циклу ***b*** необхідно задавати щораз перед початком виконання вкладеного циклу (блок 5), тобто усередині зовнішнього циклу. Якщо це буде зроблено одночасно із завданням початкового значення для параметра зовнішнього циклу ***a*** (блок 3), то внутрішній цикл виконається тільки при початковому значенні ***a***. На цьому алгоритм завершить свою роботу й, отже, не будуть перебрані всі можливі комбінації значень ***a*** і ***b***.

## 9.3. Приклад виконання лабораторної роботи «Організація циклів з невідомим числом повторень»

**Завдання**. Скласти блок-схему алгоритму для обчислення значень *y* при всіх можливих значеннях *x*, що починаються з початкового *xn*, і, що змінюються із кроком *hx*. Використовувати цикл із передумовою.

Зміст звіту по лабораторній роботі.

1. Вхідні дані: $xn > 0$, $hx = 0.6$.

2. Математична модель: $y = \cos\left(\frac{x}{\pi}\right) \cdot \sqrt{e^{-0.4x}}$

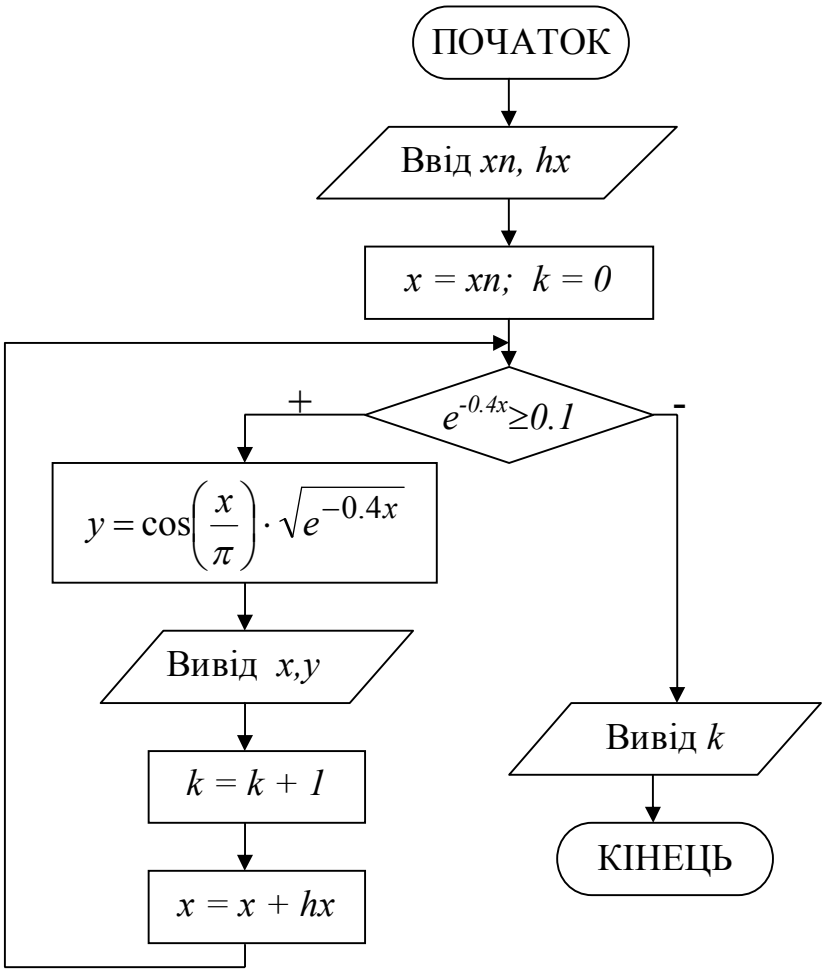
Додаткова умова завершення обчислень: обчислювати *y*, поки підкореневий вираз більше 0.1.

Визначити k – кількість обчислених *y*.

3. Обмеження: підкореневий вираз $e^{-0.4x} \geq 0$, **не перевіряти**, тому що $e^{-0.4x}$ завжди більше 0.

4. Вихідні дані: *x, y, S* і *k*.

5. Блок-схема алгоритму. Для рішення цього задачі можна використати тільки цикл із передумовою, тому що перед обчисленням *y* необхідно перевіряти умову завершення обчислень.



## 9.4. Приклад виконання лабораторної роботи «Організація вкладених циклів»

**Завдання**. Скласти блок-схему алгоритму для обчислення значень *x* і *y* при всіх можливих комбінаціях значень *a* і *b*, заданих у вигляді інтервалів від початкового до кінцевого із визначеним кроком.

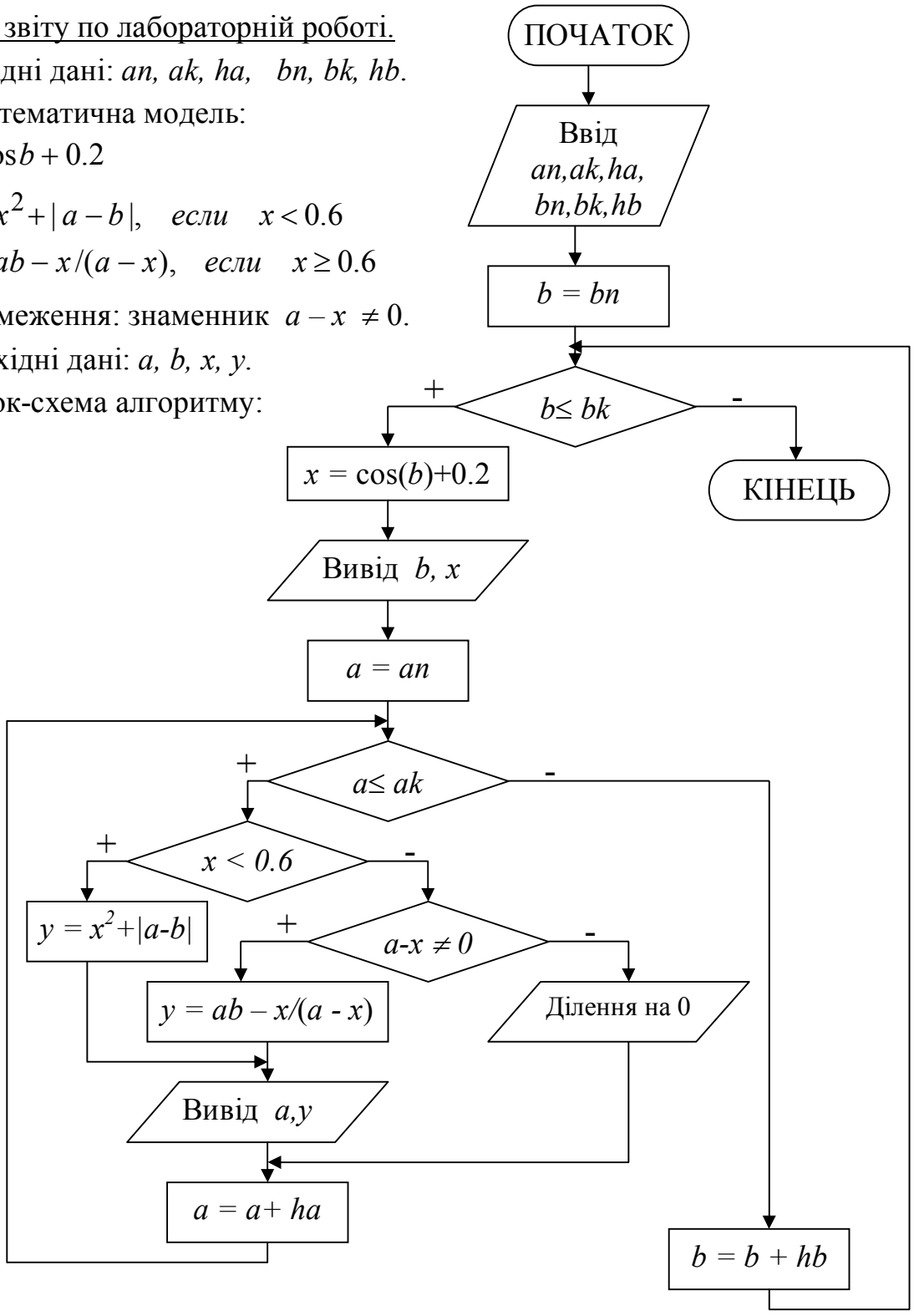
Зміст звіту по лабораторній роботі.

1. Вхідні дані: *an, ak, ha, bn, bk, hb*.
2. Математична модель:

$$x = \cos b + 0.2$$

$$y = \begin{cases} x^2 + |a - b|, & \text{если} \quad x < 0.6 \\ ab - x/(a - x), & \text{если} \quad x \geq 0.6 \end{cases}$$

3. Обмеження: знаменник $a - x \neq 0$.
4. Вихідні дані: *a, b, x, y*.
5. Блок-схема алгоритму:



## 9.5. Завдання для самостійної роботи

Скласти блок-схему алгоритму для обчислення вказаних значень при всіх можливих значеннях *x*, що починаються з початкового *xn*, і, що змінюються із кроком *hx*.

| № п/п | Необхідно обчислити | Вхідні дані | Вихідні дані |
|---|---|---|---|
| 1 | $t = 2y\ln^2 x \qquad y = \sqrt[5]{x^2 + 0.8a}$<br>Лічити t доти, поки його значення стане більше 100. Визначити k - кількість обчислених t | $a$=3.2<br>$x \geq 0.5$<br>$hx$=0.2 | *x, y, t, k* |
| 2 | $y = \frac{1}{7} + \ln^2\left(2a + \frac{x^3}{x^2+1}\right)$<br>Лічити y доти, поки вираз під знаком логарифма стане більше 0. Визначити кількість (k) обчислених y. | $a$=0.6<br>$x \leq 3$<br>$hx$= -0.4 | *x, y, k* |
| 3 | $t = a\cos^2 x - \frac{1}{3}\sqrt{4x^2 - 3x - 2}$<br>Лічити t доти, поки підкореневе вираз стане від'ємним. Визначити кількість (k) обчислених t. | $a$=1.7<br>$x \leq 5$<br>$hx$=-0.5 | *x, t, k* |

Скласти блок-схему алгоритму для обчислення вказаних значень при всіх можливих комбінаціях значень *a* і *b*, заданих у вигляді інтервалів від початкового до кінцевого із визначеним кроком.

| № п/п | Необхідно обчислити | Вхідні дані | Вихідні дані |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| 1 | $x = 5\cos(a^2 b + 0.8) + \frac{b}{(d-3)^2}$<br>$z = \frac{dx}{a-x}\lvert b^2 - d\rvert$ | $d$=1.2<br>$3 \leq a \leq 9$<br>$ha$=1<br>$-4.5 \leq b \leq 4.5$<br>$hb$=1.5 | *a, b, x, z* |
| 2 | $x = \frac{ab^2 + 2c^6}{12c + 0.5b}$<br>$Z = 4\sin^3\left(\frac{x+c}{ab}\right)\cos\left(\frac{x-c}{a^2 b}\right)$ | $c$=0.18<br>$4 \leq a \leq 8$<br>$ha$=1<br>$3.5 \leq b \leq 9.5$<br>$hb$=0.5 | *a, b, x, z* |

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 3 | $x=\frac{\lvert a-bc^2+c^3\rvert}{12(bc+1)}$<br>$y=\frac{a^2-bx^3+16cx^2}{x+a}$ | $c$=3.6<br>$1 \le a \le 7$<br>$ha$=1<br>$-0.5 \le b \le 0.5$<br>$hb$=0.1 | *a, c, x, y* |

# Лекція №10. ОДНОМІРНІ МАСИВИ

## 10.1. Основні теоретичні положення

**Масив** – це послідовність однотипних елементів, кожний з яких має одне й теж ім'я, але однозначно визначається своїм номером (**індексом**). Масив - це не скалярна величина, а структурований тип даних.

Звичайно використовуються одномірні (вектор) і двовимірні (матриця) масиви. В одномірному масиві кожний елемент має один індекс, що визначає положення елемента в масиві. У двовимірному масиві кожний елемент має два індекси. Найчастіше нумерація індексів починається з 1, але може й з 0.

| | $X_1$ | $X_2$ | $X_3$ | … | $X_N$ |
|---|---|---|---|---|---|
| Значення елемента | 2 | 0 | 1 | … | 7 |
| Індекс елемента i | 1 | 2 | 3 | … | N |

Основними характеристиками масиву є:

– розмірність, тобто кількість елементів (звичайно позначається N);

– значення елементів (наприклад, $X_1 = 2$; $X_3 = 1$ і т.д.).

Обробка масиву звичайно полягає в послідовному переборі його елементів і виконанні над ними однотипних операцій, тобто обробка масиву є циклічним обчислювальним процесом. Для цього досить організувати цикл по перебору індексів елементів масиву. Найбільше раціонально використовувати цикл «***Для***» на основі блоку модифікації (рис. 10.1).

При вході в блок модифікації (рис. 10.1. а,б, лінія потоку 1) автоматично виконуються наступні дії. Параметру циклу ***i*** присвоюється початкове значення ***in*** і перевіряється, чи не перевищує воно **кінцевого** значення ***ik***. Якщо результатом перевірки умови є істина, то відбувається перехід до виконання тіла циклу (лінія потоку 2). Після цього здійснюється повернення в блок модифікації (лінія потоку 3), збільшення параметра циклу ***i*** на значення кроку ***hi*** і перевірка умови продовження циклу й т.д. Коли поточне значення параметра циклу ***i*** перевищить кінцеве значення ***ik***, цикл завершить свою роботу (лінія потоку 4). Якщо крок зміни параметра циклу ***hi*** дорівнює 1, то його в блоці модифікації можна не вказувати (рис. 10.1. в).

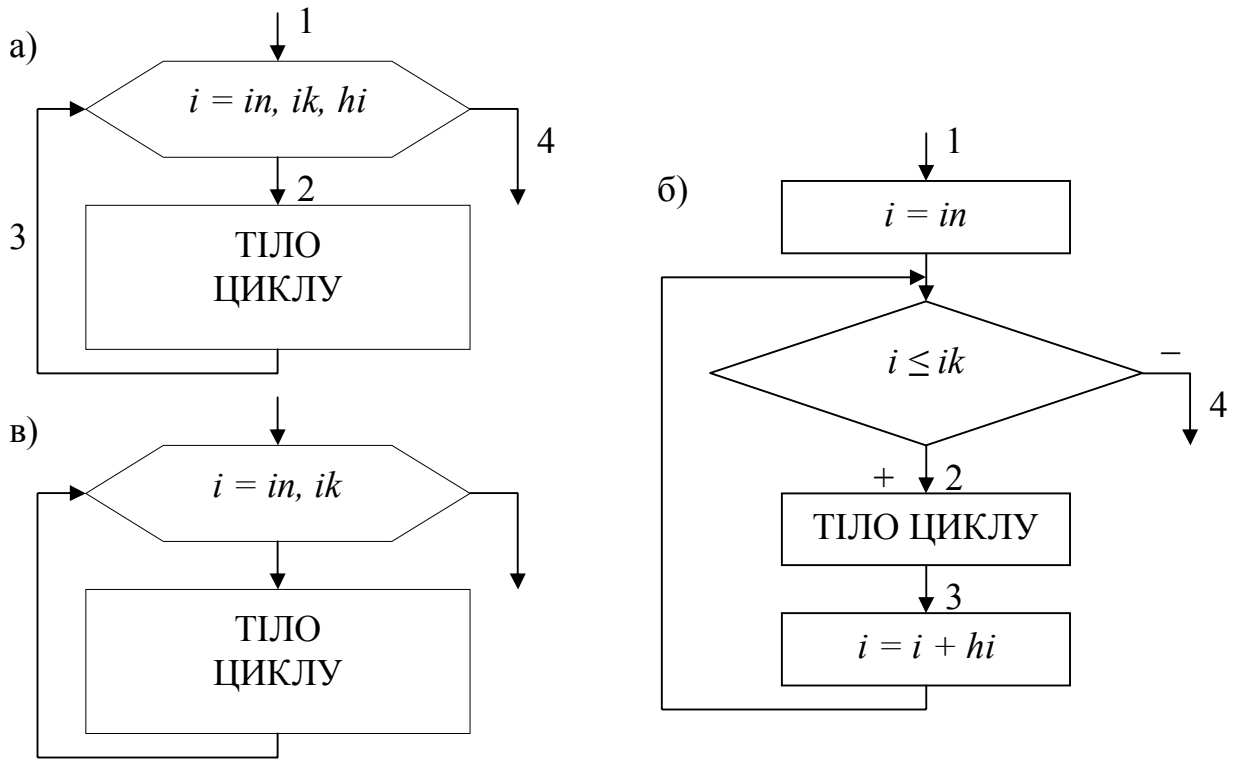



Рисунок 10.1. Цикл «Для» на основі блоку модифікації (а, в) і пояснення принципу його роботи на основі циклу із передумовою (б).

В одних мовах програмування, таких як Turbo Pascal, Delphi, існують обмеження на використання циклу «Для» (параметр циклу, його початкове й кінцеве значення повинні бути тільки цілочисленного типу; крок зміни може бути дорівнює тільки 1 або -1). В інших мовах програмування, таких як C, Visual Basic, цих обмежень немає. Але в кожному разі цикл «Для» ідеально підходить для організації обробки масивів.

## 10.2. Ввід й вивід елементів одномірного масиву

Ввід елементів масиву виконується у два етапи. Спочатку вказується його розмірність (кількість елементів), а потім задаються значення для кожного елемента масиву. Розглянемо два способи організації вводу елементів одномірного масиву.

**1 спосіб.** Алгоритм вводу масиву X розмірністю $N$ ($i = 1 \div N$) (рис. 10.2. а). На першому етапі вводиться розмірність масиву $N$ (блок 1). На другому етапі організовується цикл «Для» на основі блоку модифікації, що виконує заелементне введення довільних однотипних значень компонент масиву.

Параметр циклу $i$ у теж час є індексом елементів масиву X. Блок модифікації (блок 2) перебирає значення $i$ від 1 до $N$ і для кожного значення $i$ виконується введення значення для $X_i$-го елемента масиву (блок 3), тобто

при $i = 1$ буде введене значення для $X_1$,

при $i$ =2 – для $X_2$ і т.д.

Таким чином, за один крок виконання циклу вводиться один елемент масиву X.

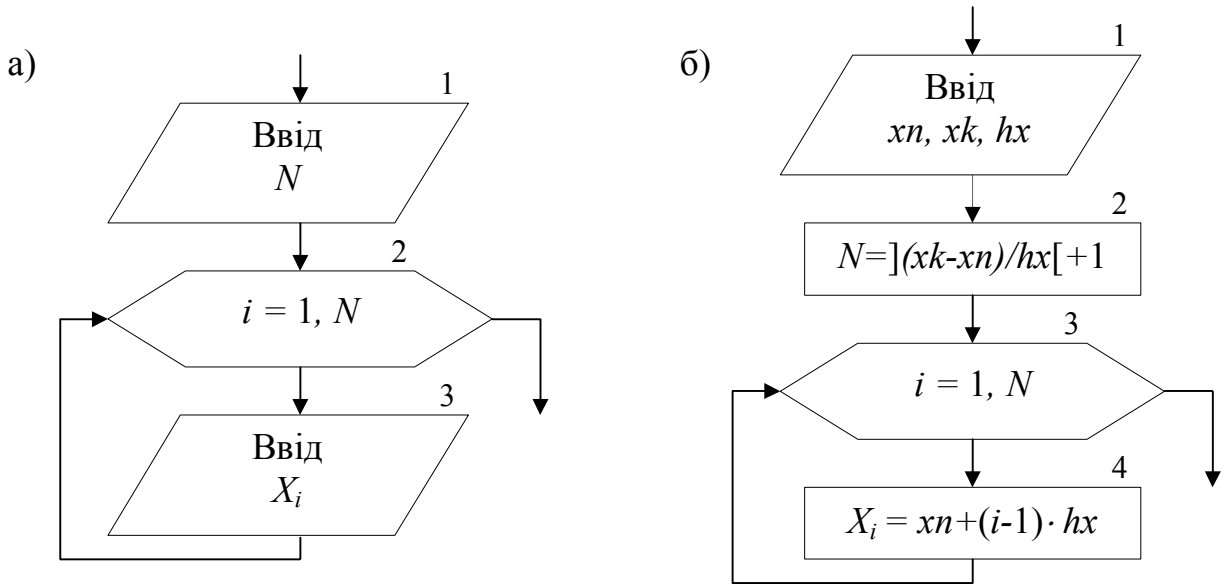



Рисунок 10.2. Заелементне введення масиву (а) і обчислення елементів масиву, значення яких лежать у заданому інтервалі (б).

**2 спосіб.** Алгоритм формування масиву X, значення елементів якого належать заданому інтервалу $xn \le X_i \le xk$ із кроком зміни $hx$ (рис. 10.2 б). На першому етапі вводяться значення $xn$, $xk$ і $hx$ (блок 1) і обчислюється $N$ – розмірність масиву X (блок 2). На другому етапі організовується цикл «Для» на основі блоку модифікації (блок 3), у якому перебираються значення $i$. На кожному кроці циклу обчислюється значення одного елемента масиву X (блок 4), тобто

при $i$ =1 буде обчислене значення для $X_1=xn+(1-1)\cdot hx=xn$,

при $i$ =2 – для $X_2= =xn+(2-1)\cdot hx=xn+hx$ і т.д.

Вивід масиву також виконується заелементне за допомогою циклу «Для» (рис 10.3).

Цикли по вводу або виводу елементів масивів не обов'язково робити окремо. Їх можна об'єднувати із циклами по обробці елементів масивів.

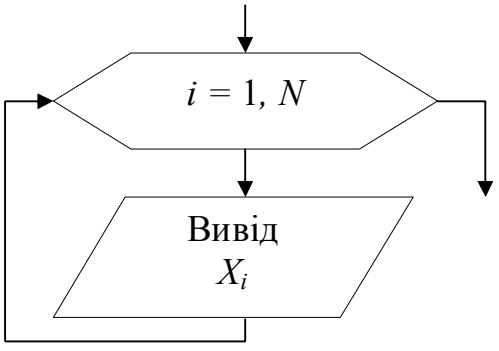



Рисунок 10.3. Вивод елементів масиву

**Приклад 10.1.** Використовуючи елементи вхідного масиву X розмірності ***N***, обчислити елементи масиву Y по наступній формулі: $Y_i=1/X_i$. Визначити середнє арифметичне додатних елементів масиву Y.

Результуючий масив Y буде мати таку ж розмірність, що й вхідний масив X.

Рішення поставленої задачі можна розділити на 4 етапи: введення елементів вхідного масиву X, обчислення елементів масиву Y, обчислення середнього арифметичного додатних елементів масиву Y і вивід елементів масиву Y. Для реалізації кожного етапу необхідно організувати окремі цикли для перебору елементів масивів. Однак, такий спосіб рішення не раціональний, хоча й можливий. Внаслідок того, що організація всіх чотирьох циклів буде цілком ідентична, етапи рішення з 2-го по 4-й можна реалізувати за допомогою тільки одного циклу «Для». Це значно зменшить розмір алгоритму й швидкість виконання програми. Блок-схема алгоритму наведена на рис. 10.4.

Введення вхідного масиву (блоки 2-4) здійснюється заелементне, способом, який описано вище. У блоці 5 присвоюються початкові значення для змінних ***S*** і ***k*** (сума й кількість додатних елементів масиву Y). Потім організовується цикл «Для» (блок 6), на кожному кроці якого будуть виконуватися наступні дії: обчислення по заданій формулі поточного елемента масиву Y (блоки 7-9); перевірка – чи не є елемент $Y_i$, якій обчислено, додатним (блок 10) ; додавання додатного значення $Y_i$ до суми ***S*** і збільшення лічильника таких елементів ***k*** на 1 (блок 11) ; вивід поточного елемента масиву Y (блок 12).

Особу увагу варто звернути на наявність можливої «аномалії» в виразу, який обчислюється ($X_i$ не повинне бути рівним 0). **Обробка «аномалій»**, що виникають при обчисленні елементів масиву, небагато відрізняється від запропонованої в п. 7.3. Якщо просто пропустити всі операції, які неможливо виконати, то поточний елемент масиву не буде обчислений. У результаті може бути сформований масив, у якого частина елементів буде не визначений (масив з «дірами»). Подальше використання такого масиву неприпустимо. Тому у випадку виникнення аномалії при обчисленні поточного елемента масиву, йому треба присвоїти таке значення, що не вплине на подальші обчислення, наприклад 0 (блок 9).

Таким чином, за допомогою одного циклу «Для» будуть по черзі перебрані всі елементи масиву X, для кожного елемента $X_i$ буде обчислено і виведено відповідний елемент масиву $Y_i$. Після виходу із циклу обчислюється й виводиться середнє арифметичне значення додатних елементів масиву Y (блоки 13-15). При цьому попередньо в блоці 13 перевіряється можливість ділення на нуль.

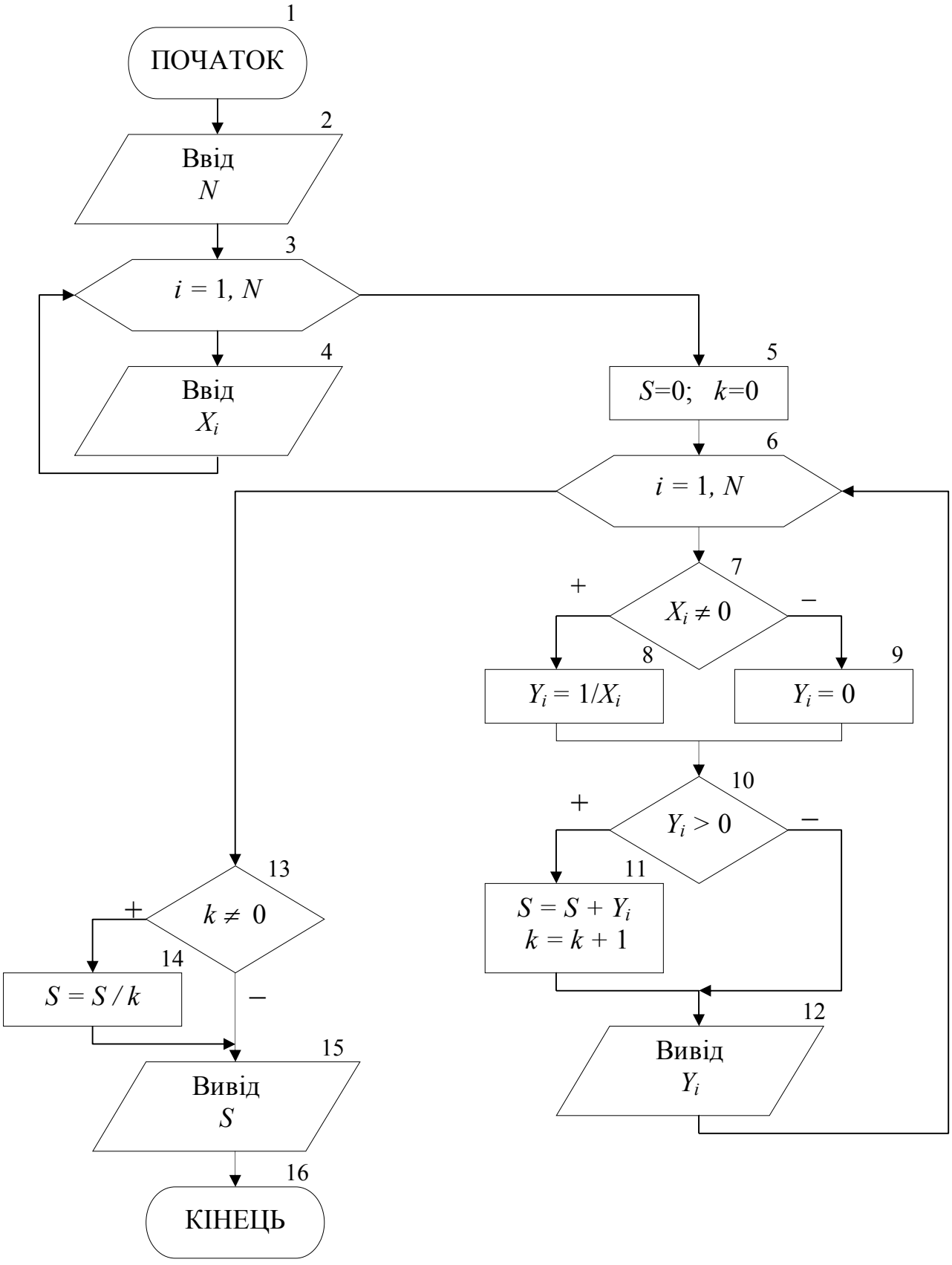



Рисунок 10.4. Блок-схема алгоритму для прикладу 10.1

**Приклад 10.2.** Використовуючи вхідний масив X, значення елементів якого належать заданому інтервалу $xn \le X_i \le xk$ із кроком зміни ***hx***, обчислити елементи масиву Y по наступній формулі: $Y_i=\sin(X_i)$. Визначити порядковий номер і значення першого додатного елемента масиву Y.

Як і в попередньому прикладі, рішення поставленої задачі можна розділити на два етапи (рис. 10.5).

На першому етапі виконується формування вхідного масиву X. Для цього вводяться граничні значення заданого інтервалу – значення ***xn***, ***xk*** і ***hx*** (блок 2) і обчислюється ***N*** – розмірність масиву X (блок 3). Потім організовується цикл «Для» на основі блоку модифікації (блок 4), на кожному кроці якого буде обчислюватися й виводитися значення ***i-го*** елемента масиву X (блоки 5-6).

При пошуку серед елементів масиву певного значення досить знайти номер елемента, у якому зберігається це значення. Тому на другому етапі для визначення першого додатного елемента в масиві Y введемо змінну *k*, що буде використовуватися для зберігання номера шуканого елемента (блок 7). Після цього організовується цикл «Для» (тіло циклу – блоки 8-13), у якому виконується обчислення по заданій формулі й вивід поточного елемента масиву Y (блоки 9-10). У підсумку кожному елементу масиву X буде відповідати елемент масиву Y.

Якщо обчислений елемент $Y_i$ є додатним (блок 11), то в блоці 12 виконується перевірка – чи є він першим таким елементом у масиві, і у випадку позитивного результату перевірки в змінної *k* зберігається номер цього елемента (блок 13). При подальшій роботі циклу у випадку, коли буде зустрічатися черговий додатний елемент масиву Y, буде виконуватися умова $k \neq 0$, і такий елемент оброблятися не буде. Таким чином, у змінної *k* буде збережений номер першого додатного елемента масиву Y.

Після завершення формування масиву Y (вихід із циклу) виконується перевірка – чи знайдено в масиві необхідний елемент (блок 14) і виводяться його номер ***k*** і значення $Y_k$ (блок 15). У противному випадку масив Y не містить додатних елементів, про що й виводиться відповідне повідомлення (блок 16).

## 10.3. Приклад виконання лабораторної роботи «Обробка одномірних масивів»

**Завдання**. Скласти блок-схему алгоритму, що на основі елементів вхідного масиву X, обчислює елементи масиву Y.

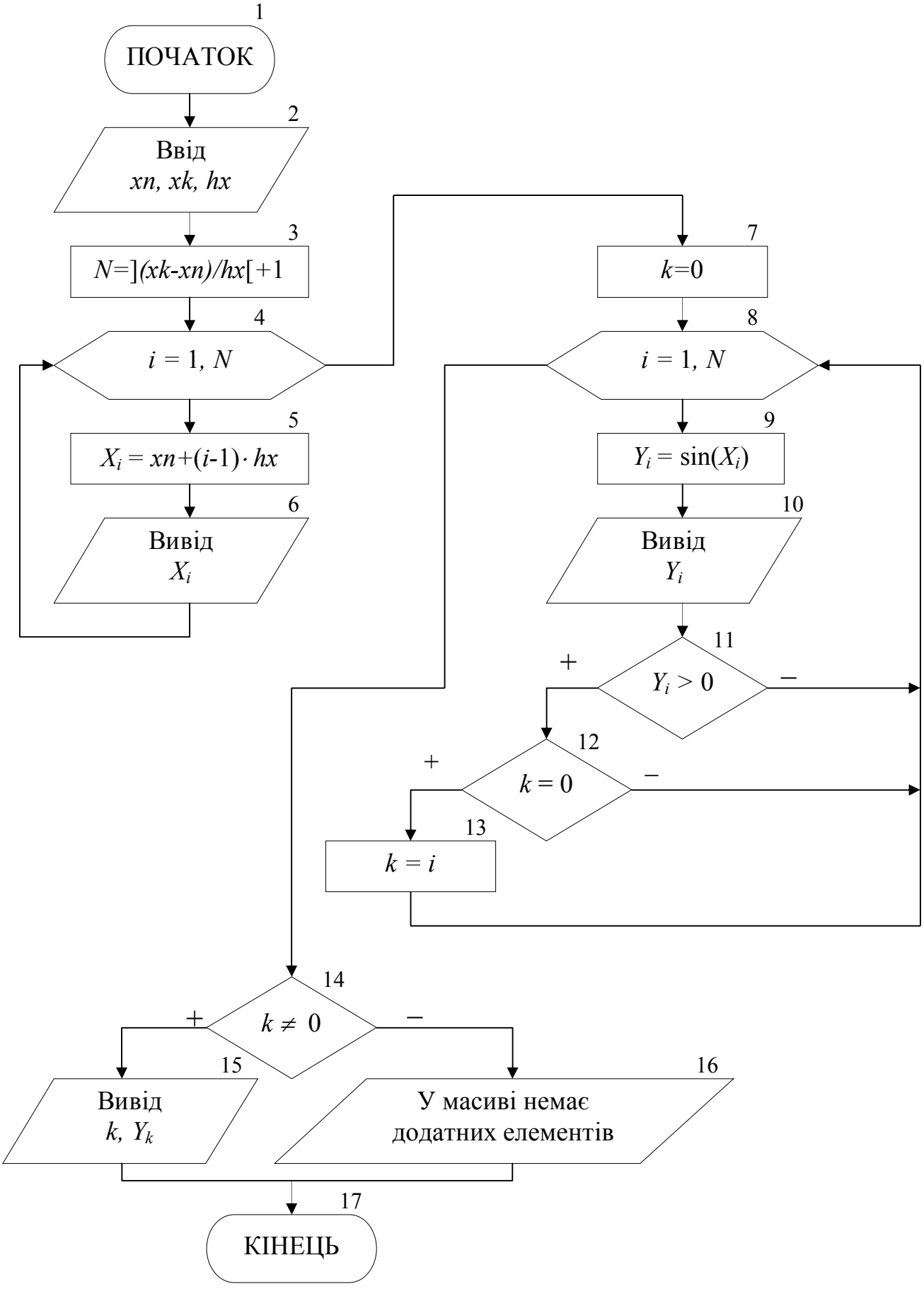



Рисунок 10.5. Блок-схема алгоритму для прикладу 10.2

Зміст звіту по лабораторній роботі.

1. Вхідні дані: масив X, розмірністю $i = 1 \div N$.

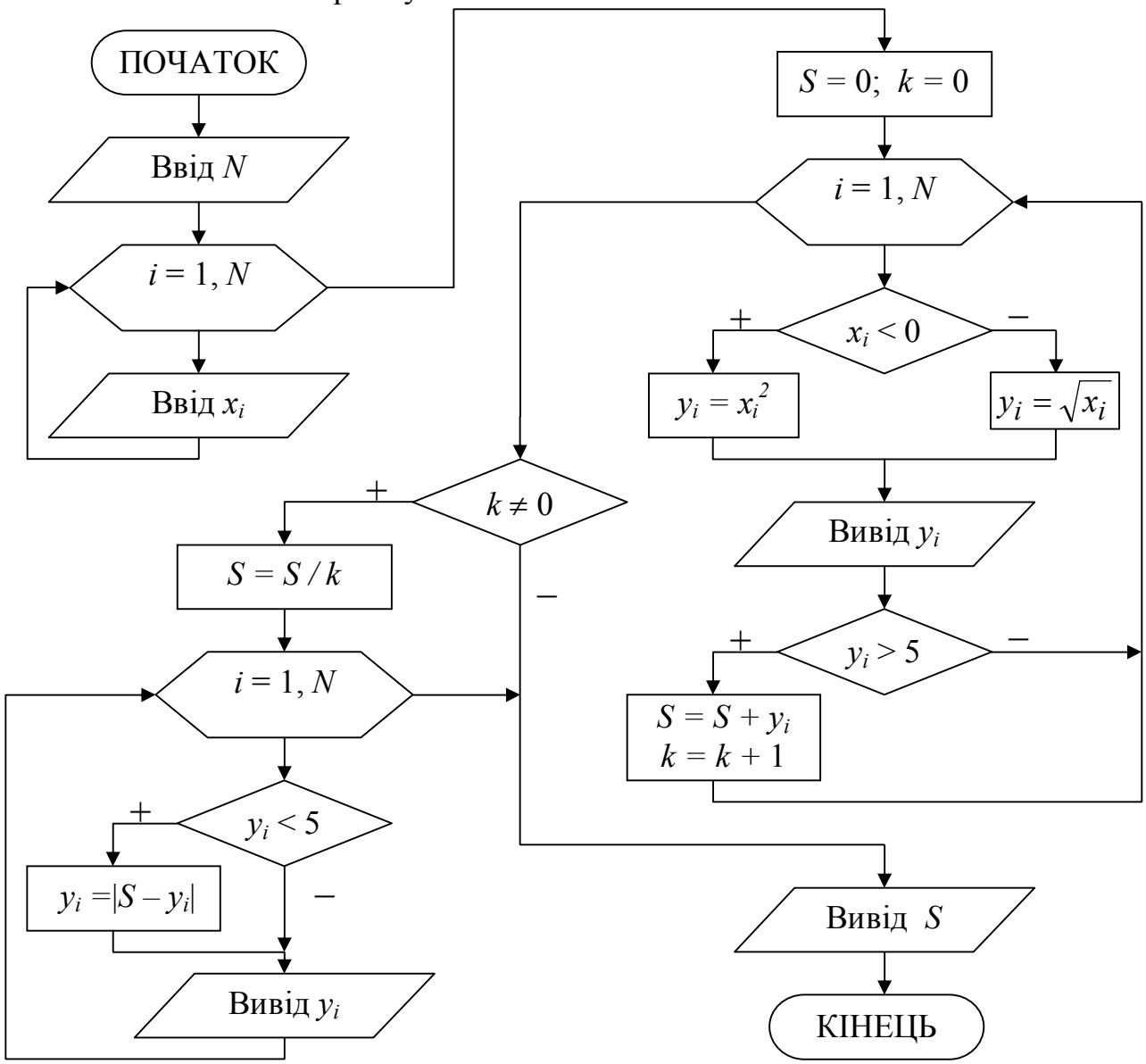
2. Математична модель:

$$y_i = \begin{cases} x_i^2, \text{ если } x_i < 0 \\ \sqrt{x_i}, \text{ если } x_i \geq 0 \end{cases}$$

Обчислити $S$ – середнє арифметичне значення елементів масиву $y_i > 5$. Замінити всі елементи $y_i < 5$ їх відхиленням від $S$.

3. Вихідні дані: масив Y, $S$.

4. Блок-схема алгоритму:




## 10.4. Завдання для самостійної роботи

**Завдання**. Скласти блок-схему алгоритму, що на основі елементів вхідного масиву, обчислює елементи вихідного масиву.

| № п/п | Модель | Вхідні дані | Вихідні дані |
|---|---|---|---|
| 1 | $y_i = \begin{cases} 4x_i^{0.6} - 2\sqrt{x_i}, & якщо\ 1 \le x_i \le 10 \\ 0.5x_i + 1, & якщо\ x_i > 10 \\ 100x_i^2 - 5e^{x_i}, & якщо\ x_i < 1 \end{cases}$<br>Середнє арифметичне (A) масиву Y і кількість $y_i > A$. | Масив $X$<br>$0.2 \le x_i \le 0.8$<br>$hx_i = 0.1$ | Масив $Y$.<br>$A$. |
| 2 | $x_i = \cos^2 z_i^2 - \sin^2 z_i$<br>$y_i = \begin{cases} 1 + e^{\sqrt{0.5x_i + 5}}, & якщо\ x_i \ge 0 \\ 1 + 0.6x_i, & якщо\ x_i < 0 \end{cases}$<br>$(x_i, y_i)$ - координати точок на площини.<br>Визначити кількість точок, розташованих в 1 і 3 квадрантах площини XOY. | Масив $Z$<br>$0.2 \le z_i \le 2.4$<br>$hz_i = 0.2$ | Масиви $X$, $Y$. |
| 3 | $m_i = \begin{cases} \dfrac{\sqrt{2y_i}\sin\dfrac{\pi}{2}y_i}{y_i + e^{y_i}}, якщо\ y_i > 1.5 \\ 2y_i - \sqrt{e^{y_i}}, якщо\ y_i \le 1.5 \end{cases}$ | Масив $Y$<br>$2.2 \le y_i \le 3.8$<br>$hy_i = 0.1$ | Масив $M$.<br>Відсоток $>0$, $<0$ і $=0$ елементів масиву M. |
| 4 | $p_i = \begin{cases} \sin^2 x_i - \cos(x_i - \pi), якщо\ x_i < \pi \\ \dfrac{2\sqrt{x_i} - \sqrt[5]{x_i}}{x_i + 2.5}, якщо\ x_i \ge \pi \end{cases}$<br>Кожен елемент $p_i$ замінити його відхиленням від середнього арифметичного елементів масиву P. | Масив $X$<br>$i = 1 \div N$ | Масив $P$ до й після заміни. Середнє арифметичне масиву P після заміни. |
| 5 | $y_i = \sin^2 x_i + \sqrt{1 + \cos^2 x_i^2}$<br>$z_i = \begin{cases} \ln(\cos^2\dfrac{\pi}{4}x_i + 0.01), якщо\ \lvert y_i \rvert > x_i^2 \\ 1 + x_i - x_i^2, якщо\ \lvert y_i \rvert \le x_i^2 \end{cases}$<br>Порядковий номер і значення останнього додатного числа в масиві Z. | Масив $X$<br>$i = 1 \div N$ | Масиви $Z$, $Y$. |

# Лекція №11. ЗАДАЧІ ОБРОБКИ ОДНОМІРНИХ МАСИВІВ

## 11.1. Знаходження максимального та мінімального елементів масиву

Принцип пошуку максимального або мінімального елемента масиву полягає в наступному: у додаткову змінну заноситься значення першого елемента масиву, що приймається за максимум (мінімум); потім організовується перебір елементів масиву, що залишилися, кожний з яких порівнюється з максимумом (мінімумом); якщо поточний елемент масиву виявляється більше (менше), чим прийнятий за максимум (мінімум), те цей елемент стає максимальним (мінімальним). Таким чином, після завершення перебору елементів масиву в додатковій змінній виявиться максимальне (мінімальне) значення серед елементів масиву. Фрагмент блок-схеми алгоритму, що реалізує описаний метод, наведений на рисунку 11.1.

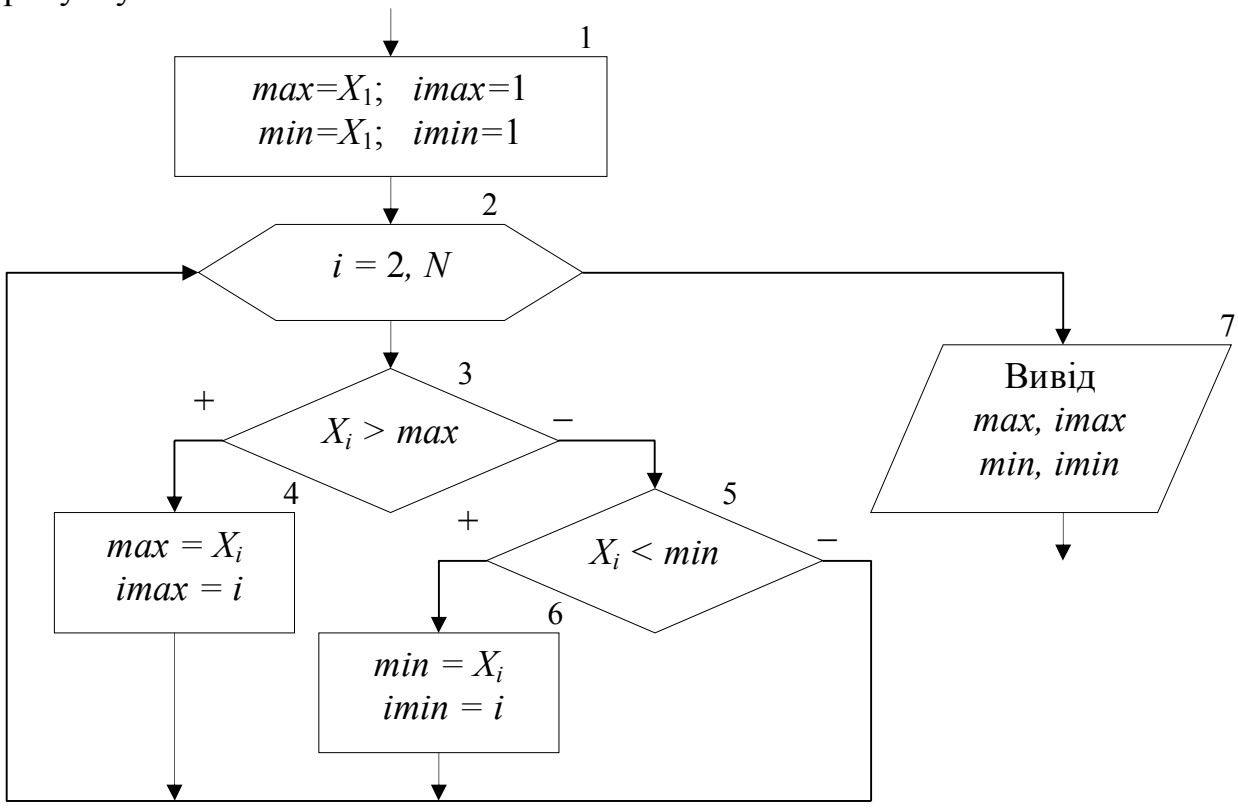



Рисунок 11.1. Пошук максимуму (мінімуму) у масиві

У блоці 1 додатковим змінним ***max*** і ***min***, у яких будуть зберігатися максимальне й мінімальне значення відповідно, присвоюється значення 1-го елемента масиву. Крім цього введені ще дві змінні ***imax*** і ***imin***, які будуть використовуватися для зберігання номерів максимального й мінімального елементів масиву. За допомогою блоку модифікації (блок 2) організовується цикл по перебору елементів масиву X (перебір починається з 2-го елемента, тому що 1-й елемент уже оброблений у блоці 1). Якщо поточний елемент масиву $X_i$ більше значення, що зберігається в змінної ***max*** (блок 3), то за максимум приймається

значення цього елемента: ***max*=*X*$_i$**, і запам'ятовується його номер: ***imax*=*i*** (блок 4). Якщо поточний елемент масиву не більше максимуму, то перевіряється, чи не менше він мінімуму (блок 5). У випадку якщо поточний елемент $X_i$ виявиться менше мінімального значення, що зберігається в змінної ***min***, те за мінімум приймається значення цього елемента: ***min*=*X*$_i$**, і запам'ятовується його номер: ***imin*=*i*** (блок 6). Після виходу із циклу будуть виведені значення максимального й мінімального елементів, а також їхнього номери в масиві (блок 7).

Наведений вище алгоритм має один недолік – у ньому використовується дві надлишкові змінні ***max*** і ***min***. Знаючи індекс елемента масиву завжди можна отримати його значення, тому немає необхідності зберігати максимальне й мінімальне значення в окремих змінних. Оптимальний алгоритм наведений на рис. 11.2.

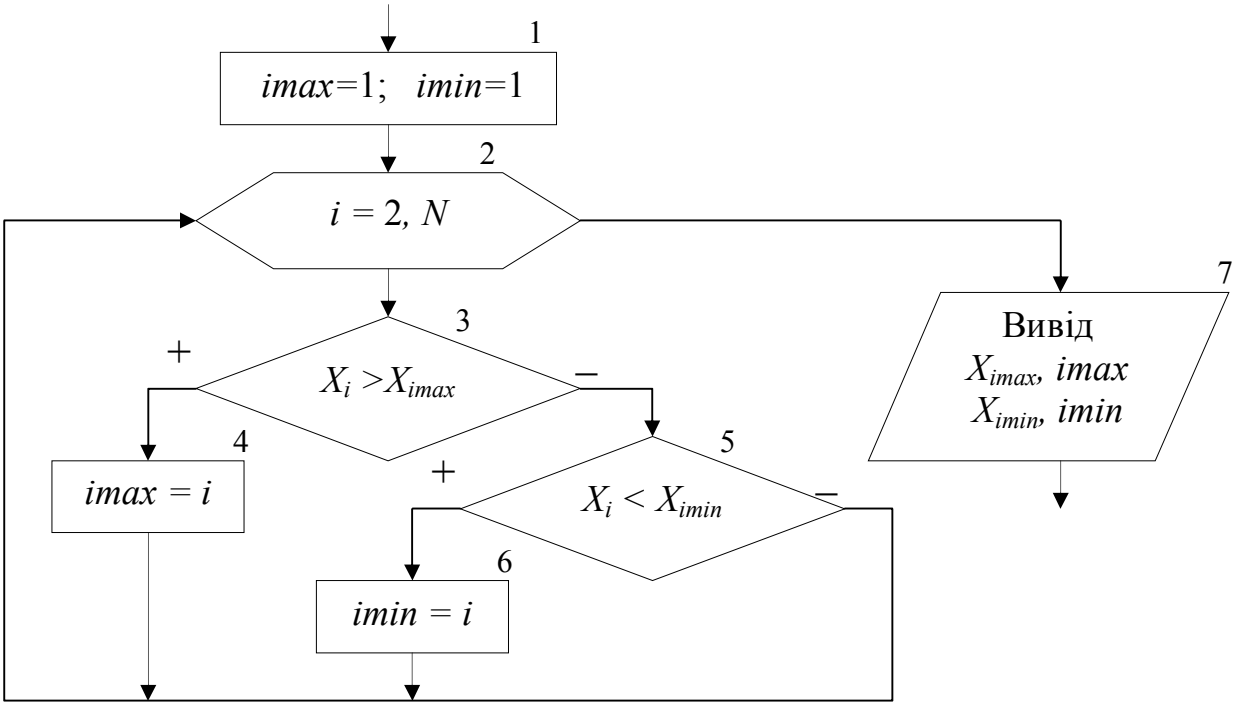



Рисунок 11.2. Оптимізація пошуку максимуму (мінімуму) у масиві

У блоці 1 додатковим змінним ***imax*** і ***imin***, які використовуються для зберігання номерів максимального й мінімального елемента, присвоюється значення 1. Таким чином, за максимум і мінімум приймається 1-й елемент масиву, саме ж значення цього елемента ніде додатково не зберігається. Елементи масиву, що залишилися, також перебираються, починаючи із другого (блок 2). При цьому кожний елемент масиву $X_i$ порівнюється з елементами, що мають номер рівний ***imax*** (блок 3) і ***imin*** (блок 6), тобто з елементами прийнятими за максимум (мінімум). Якщо результат перевірки умов є істиною, то в змінних ***imax*** і ***imin*** зберігаються індекси нових максимального (блок 4) і мінімального (блок 6) елементів. Після виходу із циклу будуть виведені значення максимального й мінімального елементів, а також їхні номери в масиві (блок 7).

Якщо в масиві є кілька елементів рівних за значенням максимальному

(мінімальному) елементу, то алгоритми, наведені на рис 11.1 і 11.2, визначать позицію тільки першого такого елемента. Наприклад, щоб знайти кількість елементів масиву рівних максимальному й позицію останнього такого елемента, можна скористатися алгоритмом, наведеним на рис. 11.3.

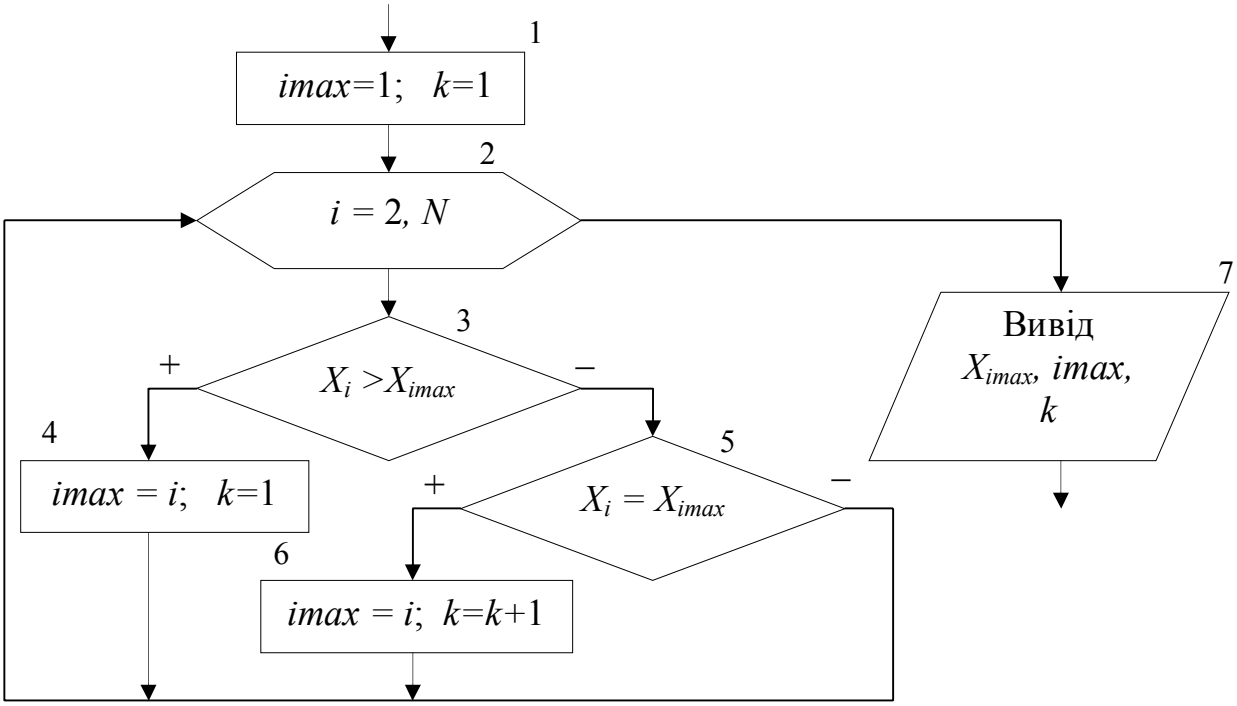



Рисунок 11.3. Визначення кількості елементів масиву рівних максимуму

На початку алгоритму за максимум приймається 1-й елемент масиву й уводиться змінна ***k*** для підрахунку кількості елементів рівних максимальному (блок 1). Потім організовується цикл по перебору елементів масиву, що залишилися (блок 2). На кожному кроці циклу виконується наступна послідовність дій.

Поточний елемент масиву $X_i$ порівнюється з максимальним (блок 3). Якщо він виявляється більше елемента, який прийнято за максимальний, то запам'ятовується номер цього елемента й змінної ***k*** присвоюється 1 (блок 4). Це означає, що зустрівся перший «новий» максимальний елемент.

Якщо поточний елемент не більше максимуму, виходить, він може бути рівним або менше максимального елемента. Тому в блоці 5 поточний елемент масиву $X_i$ перевіряється на рівність максимальному. Якщо він виявляється рівним максимуму, то знову запам'ятовується номер поточного елемента й значення змінної ***k*** збільшується на 1 (блок 6). Це означає, що зустрівся наступний елемент, який дорівнює максимальному елементу. У підсумку після виходу із циклу в змінної ***imax*** буде зберігатися індекс останнього по рахунку елемента рівного максимальному, а в змінної ***k*** кількість елементів, рівних максимуму. Якщо ж із блоку 6 забрати операцію ***imax=i***, то в змінної ***imax***, як і в попередніх

прикладах буде збережений номер першого за числом елемента рівного максимальному.

В деяких задачах неможливо спочатку прийняти за максимум (мінімум) перший елемент масиву. Наприклад, якщо елементи масиву ще не визначені (не введені або не обчислені), або якщо необхідно знайти максимум (мінімум) тільки серед додатних, парних за значенням і т.д. елементів. У другому випадку невідомо коли в масиві зустрінеться перший такий елемент. У таких ситуаціях можна використовувати спосіб, реалізований в алгоритмі із прикладу 11.1.

**Приклад 11.1.** У масиві X [1÷N] знайти мінімальний додатний елемент.

Масив X може містити як додатні, так і від'ємні елементи. Тому невідомо який елемент масиву або довільне значення можна прийняти за початковий мінімум. Рішення поставленої задачі можна розбити на два етапи: пошук позитивних елементів у масиві з визначенням першого такого елемента й пошук серед позитивних елементів мінімального за значенням. Блок-схема алгоритму наведена на рис. 11.4.

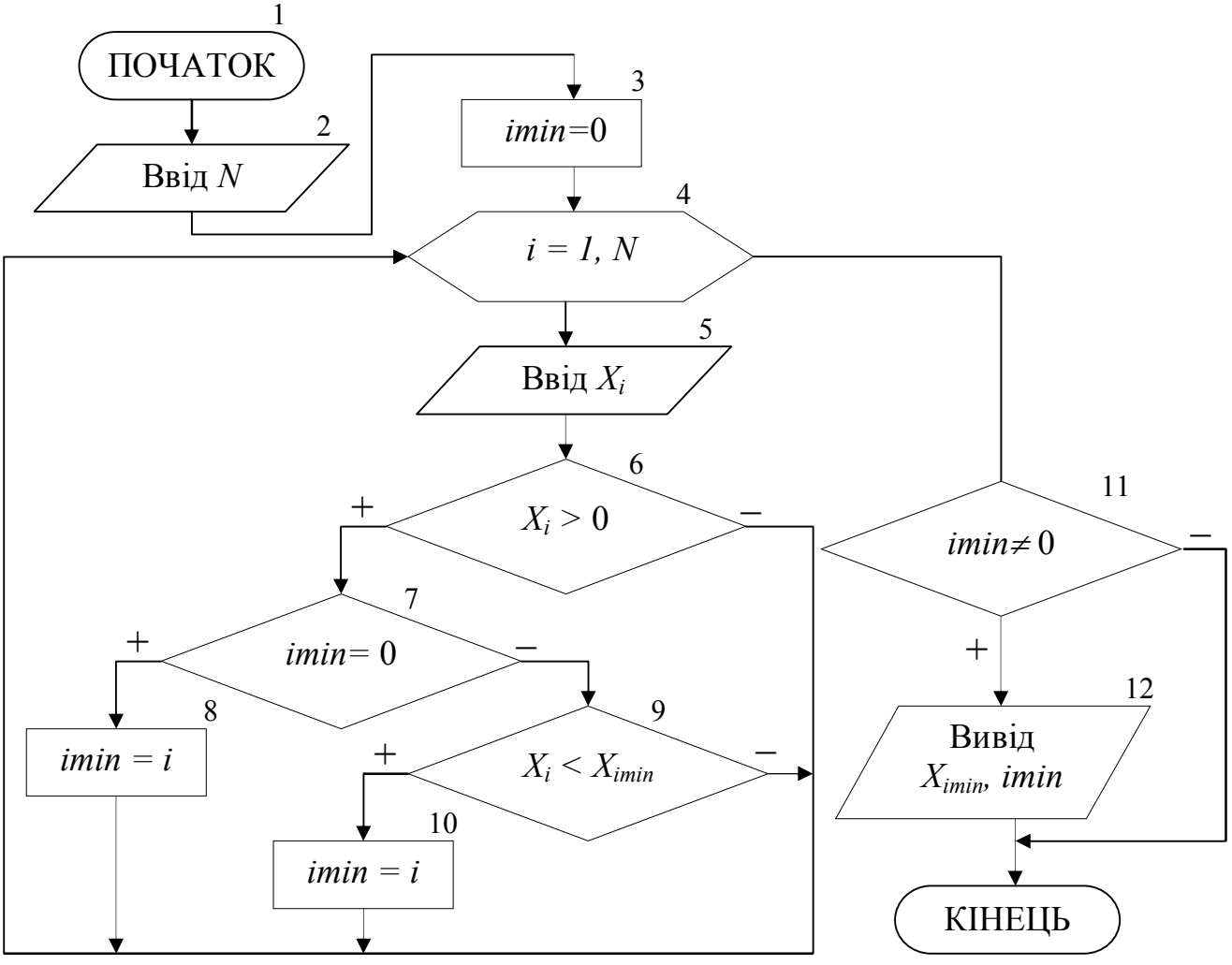



Рисунок 11.4. Блок-схема алгоритму для прикладу 11.1.

У нашому прикладі нумерація елементів масиву починається з одиниці, тому як початкове значення для змінної ***imin*** приймається 0 (блок 3), тобто значення яке не можуть приймати індекси елементів масиву X. У такий спосіб показується, що при обробці масиву ще не зустрічався додатний елемент, якого можна прийняти за початкове значення для мінімуму. Потім організовується цикл по перебору всіх елементів масиву (блок 4). На кожному кроці циклу виконується наступна послідовність дій.

Вводиться поточний елемент масиву $X_i$ (блок 5). У блоці 6 перевіряється, чи не є поточний ***елемент*** масиву $X_i$ додатним. Якщо він від'ємний або дорівнює нулю, то такий елемент пропускається й над ним не виконується ніяких дій. Якщо $X_i > 0$, то в блоці 7 перевіряється, чи не є цей елемент першим додатним елементом масиву, що зустрівся (ознакою чого ***служить*** значення ***imin*** рівне 0). У цьому випадку в блоці 8 змінної ***imin*** буде привласнене поточне значення ***i***. Таким чином, за початкове значення для мінімуму буде прийняте значення першого за числом додатного елемента масиву X. Ця ситуація може виникнути тільки один раз, і при подальшій роботі циклу блок 8 більше виконуватися не буде. Інші додатні елементи масиву в блоці 9 будуть порівнюватися з елементом масиву, який прийнято у сучасний момент за мінімум. Якщо такий елемент виявиться менше мінімального, то в блоці 10 у змінної ***imin*** буде збережений його номер ***i***.

Після виходу із циклу перевіряється, чи не дорівнює значення ***imin*** нулю (блок 11) , тому що відразу ж виводити значення мінімального елемента масиву $X_{imin}$ і його номер ***imin*** (блок 12) не можна. Це пояснюється тим що, якщо в масиві X не буде додатних елементів, то значення змінної ***imin*** залишиться рівним нулю, а звертання до неіснуючого елементу масиву $X_0$ не припустимо.

## 11.2. Сортування елементів масиву

Під сортуванням елементів масиву розуміється впорядкування елементів масиву в порядку зростання (убування) їхніх значень. Для цього необхідно організувати два вкладених цикли. Зовнішній цикл буде перебирати елементи, починаючи з 1-го до передостаннього, і вибирати так званий «головний» елемент. Внутрішній цикл буде перебирати елементи, починаючи з наступному послу поточного «головного» і до останнього.

Алгоритми сортування елементів масиву по зростанню або убуванню повністю ідентичні й відрізняються тільки умовою, яку перевіряють у внутрішньому циклі. Тому як приклад розглянемо алгоритм сортування елементів масиву по зростанню їхніх значень (рис. 11.5).

Зовнішній цикл «Для» на основі блоку модифікації (блок 1) по параметру

***i*** буде перебирати всі елементи масиву X, починаючи з першого й закінчуючи передостаннім (***N-1***-им) елементом. Для кожного обраного в зовнішньому циклі «головного» елемента $X_i$ організовується внутрішній цикл «Для» на основі блоку модифікації (блок 2), що буде перебирати елементи того ж самого масиву X, тільки по параметру ***j***. При цьому перебиратися будуть елементи з наступного (***i***+1) після поточного «головного» елемента й закінчуючи останнім (***N***-им) елементом. Таким чином, обраний ***i***-й елемент у внутрішньому циклі (блок 3) по черзі буде порівнюватися з усіма розташованими послу нього в масиві елементами (елементами з індексом ***j***). Якщо «головний» ***i***-й елемент виявляється більше, ніж наступний ***j***-й елемент, то їхні значення міняються місцями (блоки 4-6). Обмін значень двох елементів масиву виконується за 3 кроки, з використанням додаткової змінної ***a***.

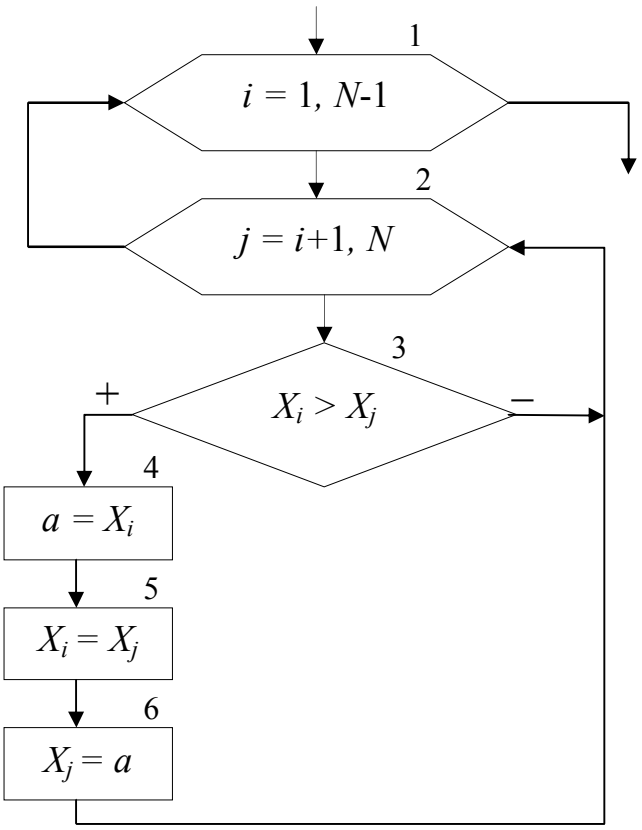



Рисунок 11.5. Сортування елементів масиву по зростанню

У процесі цих обмінів в ***i-y*** позицію будуть виставлятися всі менші й менші значення. Після завершення роботи внутрішнього циклу в ***i***-м елементі масиву буде знаходитися найменше значення, серед значень, які знаходяться після «головного» елемента. Потім у зовнішньому циклі вибирається наступний «головний» елемент, для якого виконуються ті ж самі дії й т.д. У підсумку весь масив буде відсортований у порядку зростання значень його елементів.

Нехай масив X складається із чотирьох елементів {5; 7; 3; 2}. Покрокове виконання внутрішнього циклу, при обраному в якості «головного» 1-го, а потім 2-го елементів масиву, наведено нижче:

| № кроку | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| 1 | **5** | **7** | 3 | 2 |
| | ↑***i*** =1 | ↑***j*=2** | | |
| 2 | **5** | 7 | **3** | 2 |
| | ↑***i*** =1 | | ↑***j*=3** | |
| 3 | **3** | 7 | 5 | **2** |
| | ↑***i*** =1 | | | ↑ ***j*=4** |
| Підсумок | **2** | 7 | 5 | 3 |

| № кроку | $X_1$ | $X_2$ | $X_3$ | $X_4$ |
|---|---|---|---|---|
| 1 | 2 | **7** | **5** | 3 |
| | | ↑***i*** =2 | ↑***j*=3** | |
| 2 | 2 | **5** | 7 | **3** |
| | | ↑***i*** =2 | | ↑***j*=4** |
| Підсумок | **2** | **3** | 7 | 5 |

У підсумку спочатку в 1-й елемент масиву занесений найменше значення. Потім в 2-й елемент масиву занесений найменше значення серед значень, що залишилися, й т.д.

Для того щоб цей алгоритм виконував сортування по убуванню досить у блоці 3 поміняти знак більше на знак менше.

Крім описаного вище способу існує ще кілька інших способів сортування елементів масиву, що відрізняються продуктивністю. Однак у кожному разі для виконання сортування потрібно два вкладених цикли.

## 11.3. Циклічний зсув елементів масиву

Під циклічним зсувом розуміється перестановка значень всіх елементів масиву на одну або кілька позицій уліво або вправо. При цьому значення першого (останнього) елемента масиву заноситься в останній (перший) елемент масиву залежно від напрямку зсуву.

Схематичне зображення циклічного зсуву елементів масиву вправо на одну позицію, яке виконується в три етапи, показане на рис. 11.6. Блок-схема алгоритму, що виконує такий зсув, показана на рис. 11.7.

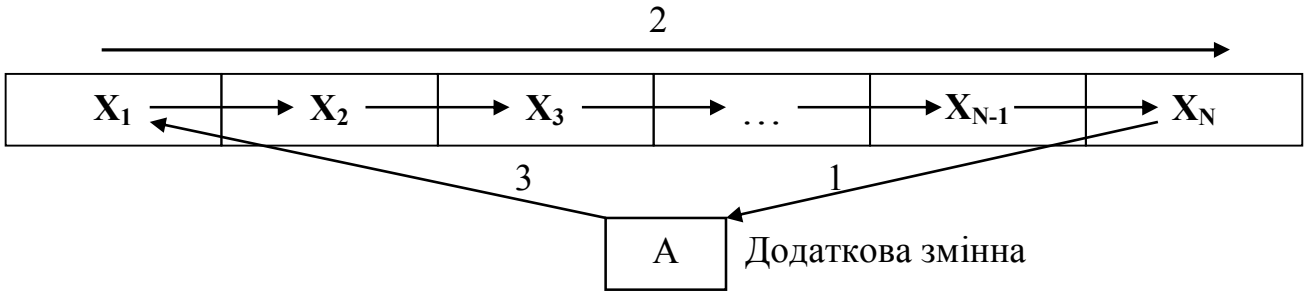



Рисунок 11.6. Принцип циклічного зсуву елементів масиву вправо

На 1-му етапі значення останнього елемента масиву $X_N$ заноситься в додаткову змінну $A$ (блок 1). На 2-му етапі організовується цикл «Для» на основі блоку модифікації (блок 2), що перебирає елементи масиву X у зворотному порядку, тобто із правого краю (крок -1), починаючи з $N$-го й закінчуючи 2-м елементом. На кожному кроці циклу поточному елементу $X_i$ присвоюється значення попереднього елемента $X_{i-1}$ (блок 3). У результаті виконання 2-го етапу значення попередньо збереженого останнього елемента масиву буде вилучено. Значення інших елементів масиву будуть зміщені на одну позицію вправо. Значення першого елемента масиву буде продубльоване в другому елементі. На 3-му етапі алгоритму значення додаткової змінної $A$, у якій зберігається останній елемент вхідного масиву, буде занесене в перший елемент масиву $X_1$. У підсумку буде виконане циклічний зсув елементів масиву X на одну позицію вправо.

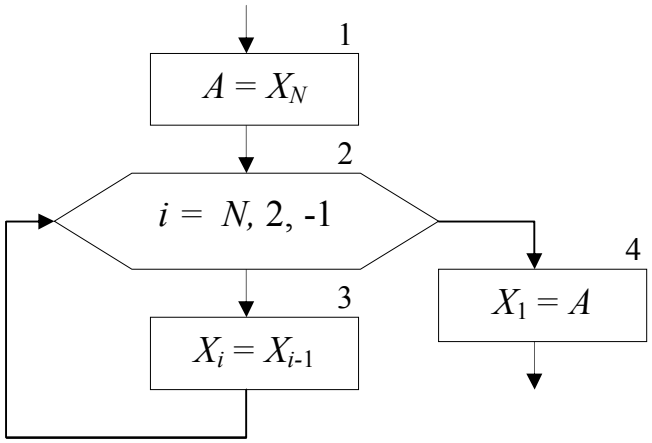



Рисунок 11.7. Циклічний зсув вправо елементів масиву

Варто звернути увагу на граничні значення параметра циклу $i$, що змінюється від $N$ до 2. При підстановці цих значень у формулу зсуву не повинне відбуватися звертання до неіснуючих елементів масиву. При $i = N$ формула зсуву приймає вид $X_N := X_{N-1}$, при $i = 2$ – $X_2 := X_1$. У випадку якби права границя параметра $i$ була рівна 1, то у формулі зсуву відбувалося б звертання до неіснуючого елемента масиву з індексом 0: $X_1 := X_0$, що неприпустимо.

Схематичне зображення циклічного зсуву елементів масиву вліво на одну позицію, яке виконується в три етапи, показане на рис. 11.8. Блок-схема алгоритму, що виконує такий зсув, показана на рис. 11.9.

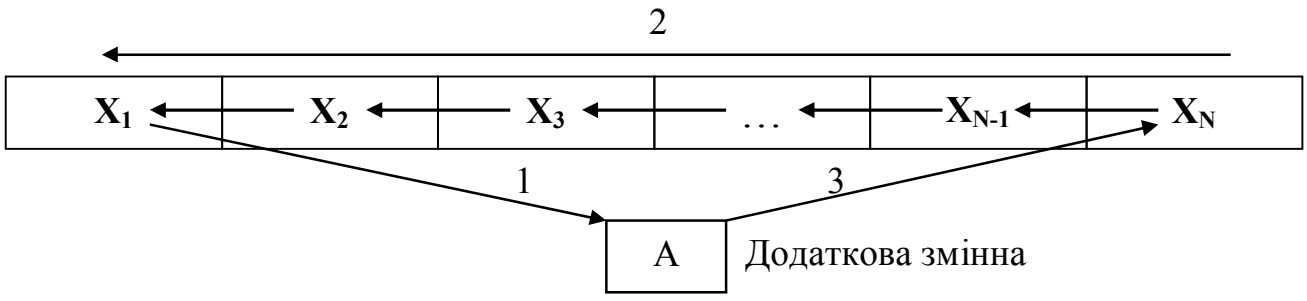



Рисунок 11.8. Принцип циклічного зсуву елементів масиву вліво

На 1-му етапі значення першого елемента масиву $\boldsymbol{X_1}$ заноситься в додаткову змінну $\boldsymbol{A}$ (блок 1). На 2-му етапі організовується цикл «Для» (блок 2), що перебирає елементи масиву X у прямому порядку, тобто з лівого краю (крок +1), починаючи з 1-го й закінчуючи ***N-1***-м елементом. На кожному кроці циклу поточному елементу $\boldsymbol{X_i}$ присвоюється значення наступного елемента масиву $\boldsymbol{X_{i+1}}$ (блок 3). У результаті виконання 2-го етапу значення попередньо збереженого першого елемента масиву буде вилучено. Значення інших елементів масиву будуть усунуті на одну позицію вліво. Значення останнього елемента масиву буде продубльоване в передостанньому елементі. На 3-му етапі алгоритму значення додаткової змінної $\boldsymbol{A}$, у якій зберігається перший елемент вхідного масиву, буде занесеної в останній елемент масиву $\boldsymbol{X_N}$. У підсумку буде виконане циклічний зсув елементів масиву X на одну позицію вліво.

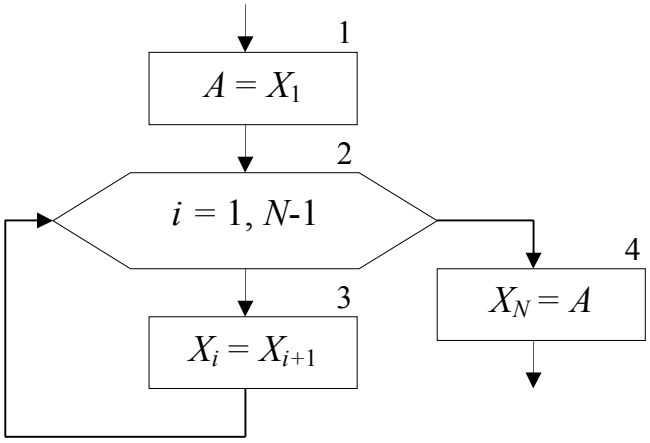



Рисунок 11.9. Циклічний зсув вліво елементів масиву

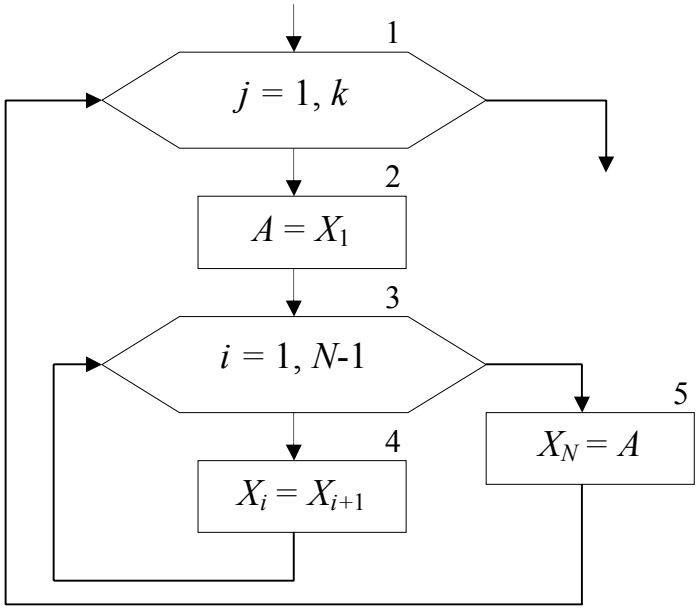



Рисунок 11.10. Циклічний зсув вліво на $\boldsymbol{k}$ позицій

Для виконання циклічного зсуву елементів масиву на $\boldsymbol{k}$ позицій досить $\boldsymbol{k}$ раз виконати алгоритм зсуву на одну позицію, тобто додати зовнішній цикл, що

буде необхідну кількість разів повторювати зсуву на одну позицію. Приклад блок-схеми алгоритму, що виконує циклічний зсуву елементів масиву на ***k*** позицій уліво, представлений на рис. 11.10.

Зсув елементів масиву знаходить застосування при видаленні або додаванні елементів у масив, а також в інших задачах.

## 11.4. Додавання й видалення елементів масиву

Для додавання нового елемента масиву необхідно виконати наступну послідовність дій: збільшити розмірність на одиницю; зрушити вправо на одну позицію частину масиву, починаючи з позиції, у яку додається елемент; в елемент, «що звільнився», записати значення, яке додається.

**Приклад 11.2.** У масив X розмірністю ***N*** додати новий елемент. Значення нового елемента й позиція, у яку він додається, указуються при вводі.

На рис. 11.11. приведений фрагмент блок-схеми алгоритму рішення поставленої задачі. Процес вводу й виводу масиву X не показаний, тому що він не відрізняється від описаного в п. 10.2.

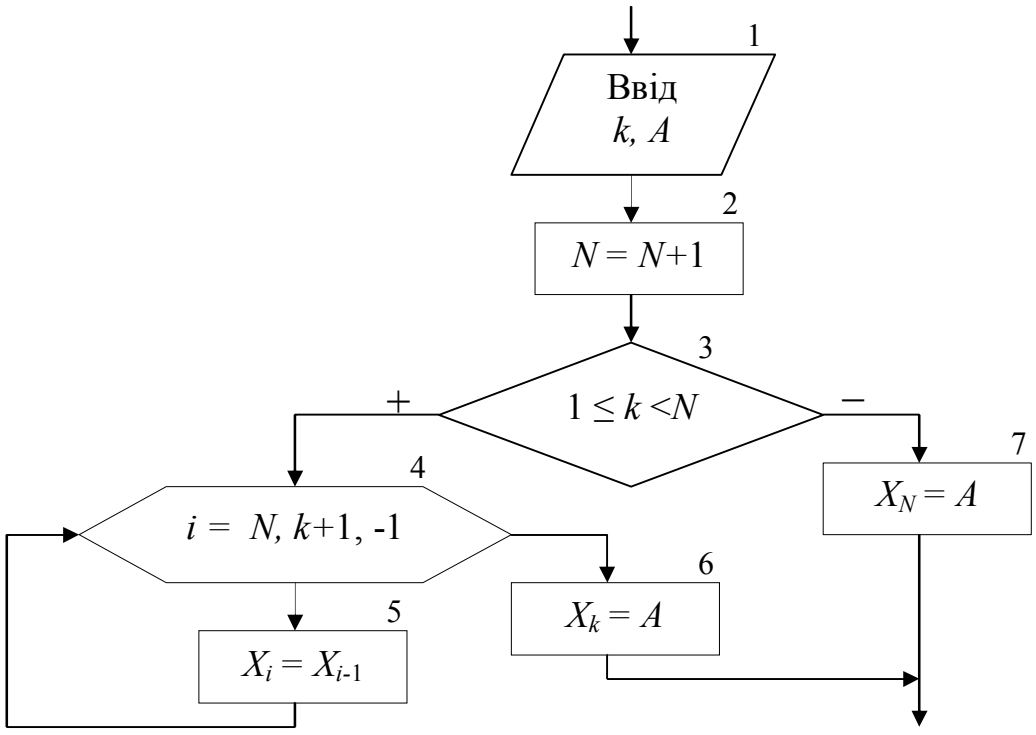



Рисунок 11.11. Додавання нового елемента масиву в k-у позицію

У блоці 1 у змінну ***A*** вводиться значення нового елемента масиву, а в змінну ***k*** заноситься номер позиції, у яку він буде додаватися. Після цього збільшується розмірність масиву ***N*** на одиницю (блок 2) і перевіряється, чи дійсно новий елемент додається в середину масиву (блок 3). У цьому випадку викону-

ється зсув частини елементів масиву X с ***N***-й до ***k***-й позиції вправо (блоки 4-5) і значення змінної ***A*** заноситься в елемент масиву, «що звільнився», з номером ***k*** (блок 6). Якщо умова в блоці 3 не виконується, то новий елемент додається в кінець масиву X (блок 7).

Для видалення елемента з масиву досить виконати зсув частини масиву, яка розташована після елементу, який видаляється, на одну позицію вліво. У результаті буде стертий елемент, що видаляється, а останній елемент масиву буде продубльоване у передостанній. Після цього досить зменшити розмірність масиву на одиницю.

**Приклад 11.3.** З масиву X розмірністю ***N*** видалити елемент із номером ***k***. Номер елемента, що видаляється, указується при вводі.

На рис. 11.12. показаний фрагмент блок-схеми алгоритму рішення поставленої задачі. При цьому процес вводу й виводу масиву X також не наводиться.

У блоці 1 у змінну ***k*** вводиться номер елемента, котрий необхідно видалити. У блоці 2 перевіряється, чи перебуває елемент, який видаляється, у середині масиву. У цьому випадку виконується зсув частини елементів масиву X с ***k-ї*** до ***N-ї*** позиції вліво (блоки 3-4) і зменшується розмірність масиву ***N*** на одиницю (блок 7). У противному випадку перевіряється, чи не є елемент, який видаляється, останнім у масиві (блок 5). Якщо це так, то для видалення елемента досить усього лише зменшити розмірність масиву ***N*** (блок 7). Якщо ж значення ***k*** не задовольняє цим двом умовам, виходить, воно задане не вірно й видалення елемента з масиву не виконується (блок 6).

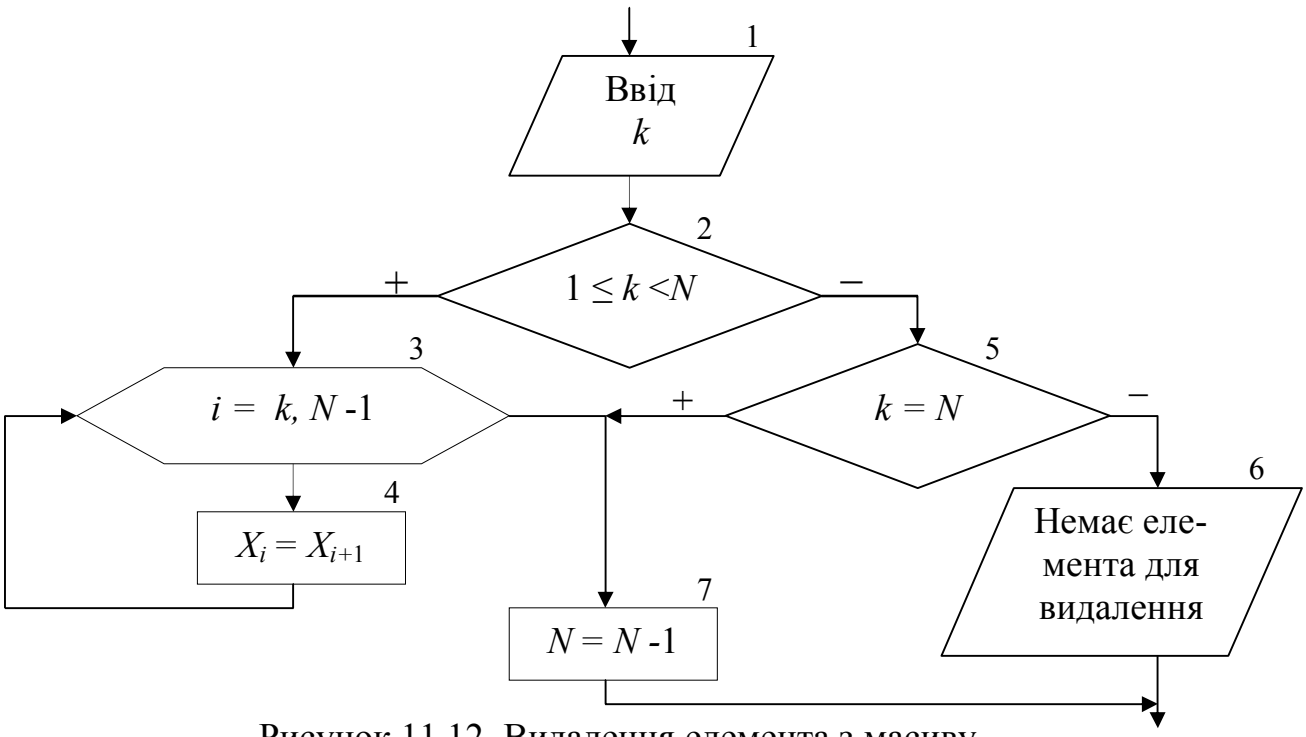



Рисунок 11.12. Видалення елемента з масиву

Іноді виникає необхідність видалити з масиву не один, а кілька елементів, що задовольняють певній умові (наприклад, всі елементи, значення яких дорівнює 0). Для цієї цілі необхідно організувати два цикли по перебору елементів масиву. Перший цикл (зовнішній) буде шукати в масиві елементи, які необхідно видалити. Другий цикл (внутрішній) буде видаляти знайдені елементи, описаним вище способом. При цьому для організації зовнішнього циклу не рекомендується використовувати цикл «Для». Це пов'язане з тим, що після видалення елемента у внутрішньому циклі, буде зменшуватися розмірність масиву, що у свою чергу є граничним значенням для параметра зовнішнього циклу.

На рис. 11.13. приводиться фрагмент блок-схеми алгоритму, у якому виконується видалення всіх нульових елементів з масиву за один цикл.

Змінна ***k*** виконує дві функції: є лічильником елементів масиву, які не треба видаляти (не рівні 0), і в теж час задає позицію в масиві X, у яку переставляється такий елемент (блок 1). У масиві здійснюється пошук (блоки 2-3) і підрахунок кількості елементів, які не повинні бути вилучені (блок 4); кожний знайдений елемент переставляється в початок масиву в позицію, що збігається з порядковим номером цього елемента (блок 4).

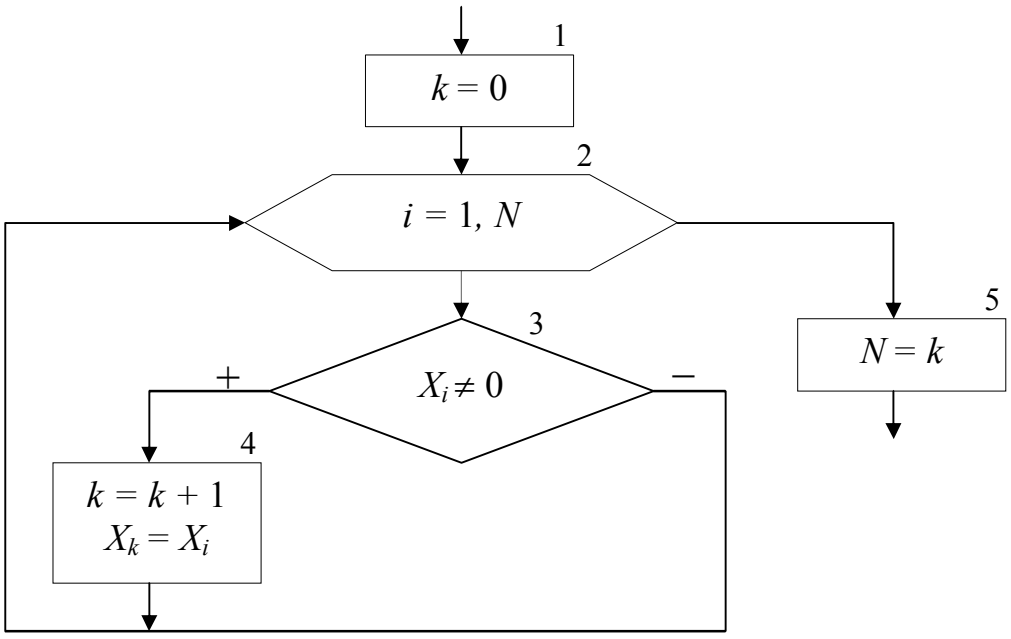



Рисунок 11.13. Видалення з масиву нульових елементів

У результаті всі нульові елементи масиву будуть пропущені, а елементи, які не рівні 0, виставлені підряд у початок масиву, розмірність якого вже буде дорівнює ***k*** (блок 5).

## 11.5. Задачі на формування масивів

При рішенні деяких задач виникає необхідність виконати формування нового масиву з елементів вхідного масиву шляхом вибору значень, що задово-

льняють певній умові відбору.

**Приклад 11.4.** Записати елементи цілочисленного масиву X розмірністю ***N*** у зворотному порядку в масив Y. Обчислити суму парних значень елементів масиву Y.

На рис. 11.14. показаний фрагмент блок-схеми алгоритму (процес введення вхідного масиву X не показаний), у якому рішення поставленої задачі виконується за допомогою одного циклу.

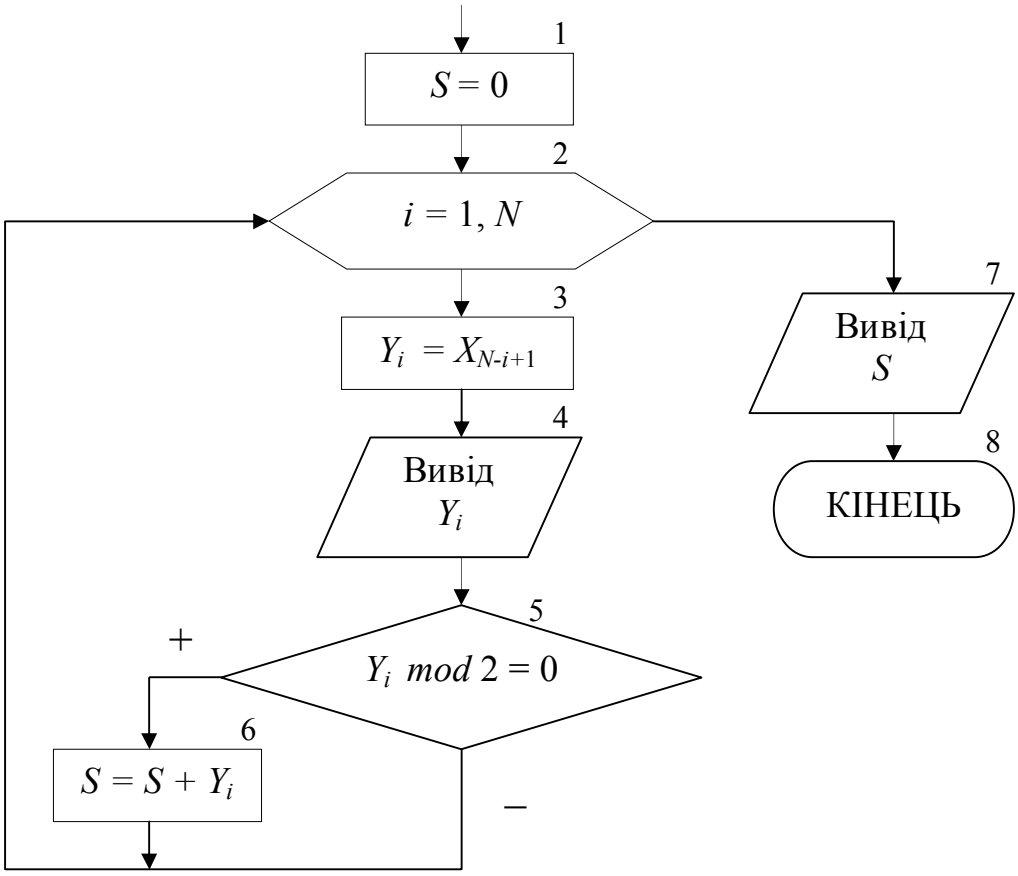



Рисунок 11.14. Блок-схема алгоритму для прикладу 11.4

У блоці 1 змінної ***S***, що буде використовуватися для зберігання суми значень елементів масиву Y, присвоюється початкове значення. Потім організовується цикл «Для» (тіло циклу – блоки 2-6). Блок модифікації перебирає елементи масиву X, починаючи з останнього елемента, і присвоює їхнього значення елементам масиву Y, які перебираються, починаючи з першого елемента (блок 3). Наприклад, при $N = 10$ будуть виконуватися наступні дії:

при $i = 1$ – $Y_1 = X_{10\text{-}1+1} = X_{10}$;

при $i = 2$ – $Y_2 = X_{10\text{-}2+1} = X_9$;

…

при $i = 10$ – $Y_{10} = X_{10\text{-}10+1} = X_1$.

У підсумку масив Y буде сформований з елементів масиву X, розташованих у зворотному порядку.

Якщо остача від цілочисленного ділення (операція ***mod***) значення поточного елемента масиву Y на 2 дорівнює 0 (блок 5), то значення цього елемента додається до суми ***S*** (блок 6).

**Приклад 11.5.** Записати додатні елементи масиву X розмірністю ***N*** у порядку їхнього проходження в масив Y. Визначити розмірність масиву Y.

На рис. 11.15. показаний фрагмент блок-схеми алгоритму (процес вводу вхідного масиву X не показаний), у якому рішення поставленої задачі виконується за допомогою одного циклу.

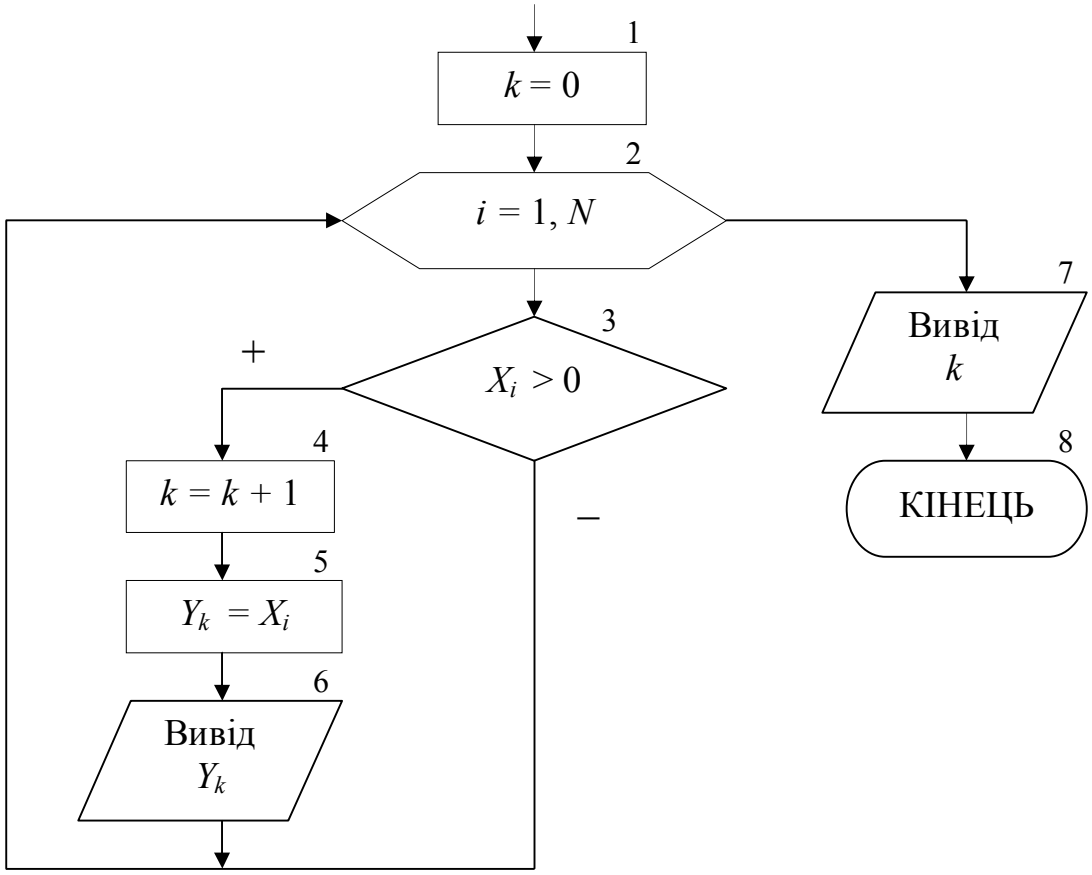



Рисунок 11.15. Блок-схема алгоритму для прикладу 11.5

Ідея алгоритму полягає в наступному. За допомогою циклу «Для» (блок 2) виконується послідовний перебір всіх елементів масиву X. Коли зустрічається додатний елемент (блок 3), те визначається його порядковий номер ***k*** (блок 4). У теж час змінна ***k*** виконує роль індексу елементів масиву Y, що дозволяє записувати додатні елементи масиву X підряд у масив Y (блок 5). Після виходу із циклу в змінної ***k*** буде зберігатися розмірність масиву Y (блок 7).

Наприклад, з масиву X={5, -3, 4, 0, -1, 3, 2} буде сформований масив Y={5, 4, 3, 2} у наступному порядку:

при $i = 1 - X_1 = 5 > 0 \rightarrow k = 0 + 1 = 1 \rightarrow Y_1 = X_1 = 5$;

при $i = 2 - X_2 = -3 < 0 \rightarrow$ дії не виконуються;

при $i = 3 - X_3 = 4 > 0 \rightarrow k = 1 + 1 = 2 \rightarrow Y_2 = X_3 = 4$;

при $i = 4 - X_4 = 0 = 0 \rightarrow$ дії не виконуються;

при $i = 5 - X_5 = -1 < 0 \rightarrow$ дії не виконуються;

при $i = 6 - X_6 = 3 > 0 \rightarrow k = 2 + 1 = 3 \rightarrow Y_3 = X_6 = 3$;

при $i = 7 - X_7 = 2 > 0 \rightarrow k = 3 + 1 = 4 \rightarrow Y_4 = X_7 = 2$.

Розмірність масиву Y – $k = 4$.

## 11.6. Приклад виконання лабораторної роботи «Обробка одномірних масивів з перестановкою елементів»

**Завдання**. Скласти блок-схему алгоритму, що на основі елементів вхідного масиву *X*, формує масив *Y*.

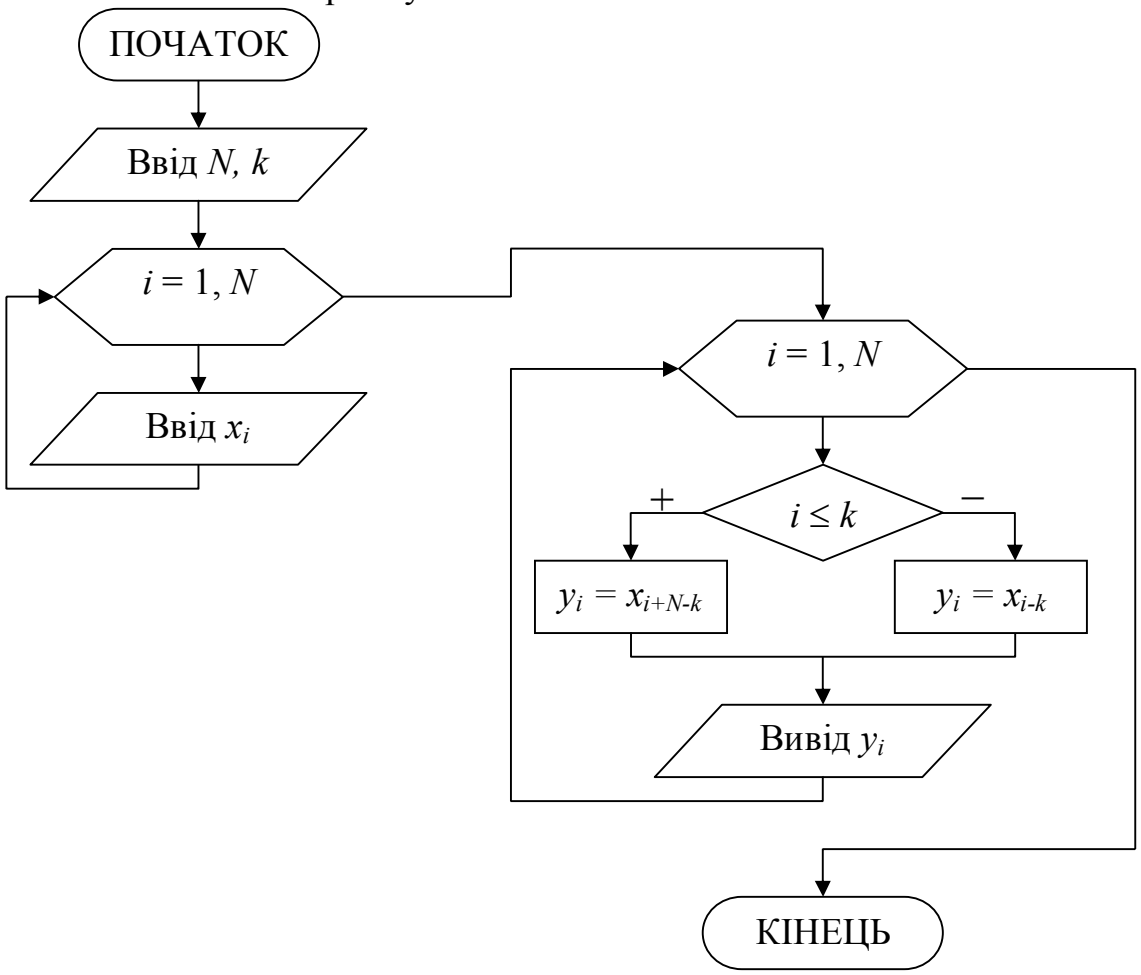
Зміст звіту по лабораторній роботі.

1. Вхідні дані: масив *X*, розмірністю $i = 1 \div N$.

2. Постановка задачі: Записати елементи масиву $X=( x_1,x_2,\ldots,x_N)$ у масив $Y=( y_1,y_2,\ldots,y_N)$, зрушивши елементи масиву *X* вправо на *k* позицій. При цьому *k* елементів з кінця масиву *X* переміщаються в початок масиву *Y*, тобто $(y_1,y_2,\ldots,y_N)=(x_{N-k+1},\ldots,x_{N-1},x_N, x_1,x_2,\ldots..,x_{N-k})$.

3. Вихідні дані: масив *Y*.

4. Блок-схема алгоритму:



### 11.7. Завдання для самостійної роботи

**Завдання**. Скласти блок-схему алгоритму, що на основі елементів вхідного масиву виконує наступні дії.

1. Визначити максимальний елемент серед додатних непарних елементів і мінімальний серед додатних парних елементів цілочисленного масиву $X=(x_1,x_2,\ldots,x_N)$.

2. Дано масив дійсних чисел $X=(x_1,x_2,\ldots,x_N)$. Записати елементи заданого масиву X у масив Y у такий спосіб: у початковій частині розташувати додатні елементи в порядку зростання, потім у порядку убування від'ємні елементи, нульові елементи не записувати.

3. У цілочисленний масив $X(1\div N)$ після кожного непарного елемента вставити максимальний елемент цього ж масиву. Визначити середнє арифметичне елементів масиву до й після вставки.

4. Обчислити середнє арифметичне елементів масиву $X=(x_1,x_2,\ldots,x_N)$, які розташовані між його мінімальним і максимальним значеннями.

5. Визначити порядкові номери й значення першого додатного і останнього від'ємного елементів цілочисленного масиву $X(1\div N)$.

6. Задано впорядкований по зростанню масив $X(1\div N)$, вставити в масив X деяке число K, зберігши впорядкованість масиву.

7. Видалити з масиву найбільший і найменший елементи. У перетвореному масиві знайти кількість найбільших елементів масиву.

8. Задано масив $X(1\div N)$. Якщо масив не є знакочергуємим, то видалити з масиву всі додатні елементи, у противному випадку – видалити від'ємні елементи.

9. Записати сім перших додатних елементів масиву $X=(x_1,x_2,\ldots,x_N)$ підряд у масив $Y=(y_1,y_2,\ldots,y_7)$. Знайти (max) – максимальний елемент масиву Y і його номер.

10. Записати елементи масиву $X=(x_1,x_2,\ldots,x_N)$, які задовольняють умові $X_i \in [2,3]$, підряд у масив $Y=(y_1,y_2,\ldots,y_k)$. Визначити (k) - кількість таких елементів.

## Лекція №12. ДВОВИМІРНІ МАСИВИ

### 12.1. Основні теоретичні положення

**Двовимірний масив** (матриця) являє собою таблицю, на перетинанні рядків і стовпців якої розташовуються елементи. Кожний елемент має два індекси. Перший індекс звичайно позначається буквою ***i*** і вказує номер рядка, у якій розташований елемент. Другий індекс позначається буквою ***j*** і вказує номер

стовпця, у якому розташований елемент (рис. 12.1). Розмірність двовимірного масиву задається двома числами: ***M*** – кількість рядків і ***N*** – кількість стовпців.

| | $j=1$ | $j=2$ | $j=3$ | ... | $j=N$ |
|---|---|---|---|---|---|
| $i=1$ | $A_{1,1}$ | $A_{1,2}$ | $A_{1,2}$ | ... | $A_{1,N}$ |
| $i=2$ | $A_{2,1}$ | $A_{2,2}$ | $A_{2,3}$ | ... | $A_{2,N}$ |
| ... | … | ... | ... | $A_{i,j}$ | ... |
| $i=M$ | $A_{M,1}$ | $A_{M,2}$ | $A_{M,3}$ | ... | $A_{M,N}$ |

Рисунок 12.1. Двовимірний масив

Двовимірний масив, у якого кількість рядків дорівнює кількості стовпців називається квадратною матрицею, у противному випадку - прямокутною.

Для обробки двовимірного масиву потрібно два вкладених цикли, при цьому найбільше зручно використовувати цикли «Для» на основі блоку модифікації. Перший буде перебирати рядки, другий – стовпці масиву. Таким чином, будуть перебрані всі елементи масиву.

Ввід двовимірного масиву (рис. 11.2), також як і одномірного виконується у два етапи.

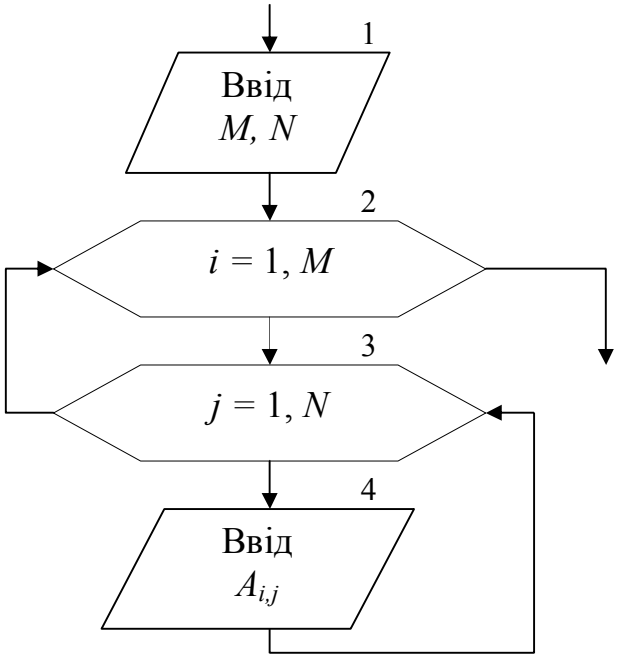



Рисунок 12.2. Ввід двомірного масиву

Спочатку вводиться розмірність масиву (блок 1), а потім значення для кожного елемента. Зовнішній цикл (блок 2) при ***i*** =1 «вибирає» 1-й рядок маси-

ву. Внутрішній цикл (блок 3) перебирає всі стовпці масиву, тобто по черзі вибираються елементи $A_{1,1}$, $A_{1,2}$, $A_{1,3}$ і т.д. до кінця 1-го рядка й вводяться їхні значення (блок 4). Після виходу із внутрішнього циклу відбувається повернення в блок 2, де вибирається 2-й рядок масиву, для якого внутрішній цикл знову перебере по черзі всі елементи $A_{2,1}$, $A_{2,2}$, $A_{2,3}$ і т.д. Таким чином, елементи двовимірного масиву будуть перебиратися по рядках.

Аналогічним образом виконується вивід елементів двовимірного масиву.

Якщо в блок-схемі на рис. 12.2. поміняти місцями параметри зовнішнього й внутрішнього циклів, тобто зовнішній цикл зробити по параметру ***j***, а внутрішній – по параметру ***i***, то елементи масиву будуть перебиратися по стовпцях.

**Приклад 12.1.** Сформувати вектор У розмірністю М, кожний елемент якого дорівнює кількості нульових елементів відповідного рядка матриці А розмірністю М на N.

Блок-схема алгоритму наведена на рис. 12.3.

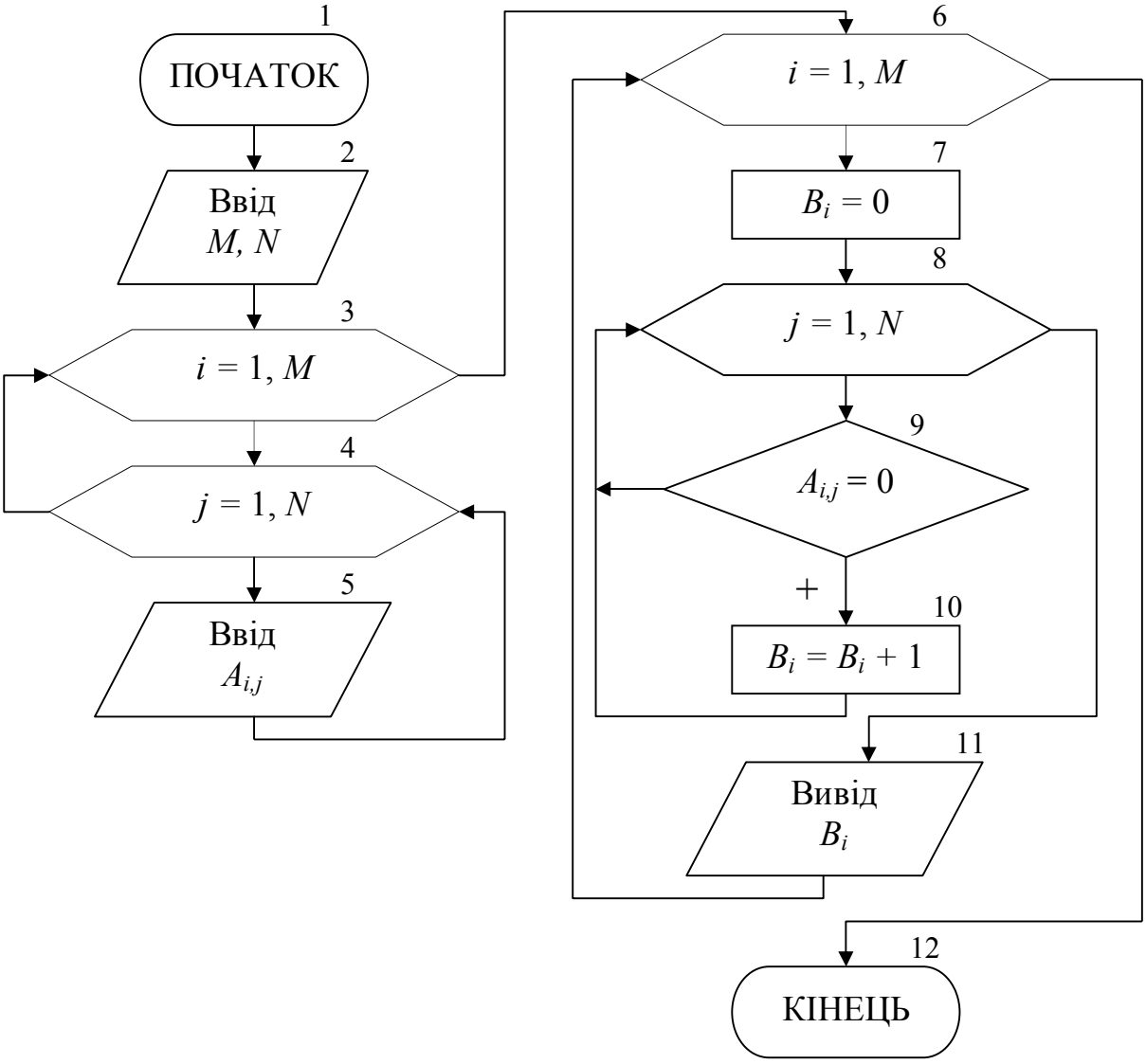



Рисунок 12.3. Блок-схема алгоритму для прикладу 12.1

Як видно з умови кількість елементів вектора B дорівнює кількості рядків матриці A. Для рішення поставленої задачі необхідно організувати порядковий перебір елементів двовимірного масиву. Зовнішнім повинен бути цикл по параметру ***i***, внутрішнім цикл по параметру ***j***. Це дасть можливість після завершення обробки кожного рядка формувати елементи одномірного масиву B.

Блоки 2-5 уводять вхідний двовимірний масив A способом, що описано вище. Потім організовується зовнішній цикл «Для» по параметру ***i*** на основі блоку модифікації (блок 6), що буде одночасно перебирати рядки масиву A и елементи одномірного масиву B. З метою оптимізації алгоритму не вводиться окрема змінна для зберігання кількості нульових елементів у кожному рядку матриці. Замість її будуть використовуватися безпосередньо елементи масиву B.

Для обраного в зовнішньому циклі ***i***-го рядка обнуляється ***i***-й елемент масиву B (блок 7). Потім організовується внутрішній цикл «Для» по параметру ***j*** (блок 8), що перебирає стовпці масиву A, тобто елементи ***i***-го рядка. Кожний елемент перевіряється на рівність нулю (блок 9), і у випадку виконання умови відбувається збільшення лічильника нульових елементів ***i***-го рядка, значення якого зберігається в елементі $B_i$ (блок 10). Після завершення обробки рядка (виходу із внутрішнього циклу) відбувається вивід ***i***-го елемента масиву B (блок 11) і перехід до наступного рядка. Обробивши всі рядки двовимірного масиву A, алгоритм завершить свою роботу.

**Приклад 12.2.** Сформувати вектор B розмірністю N, кожний елемент якого дорівнює середньому арифметичному значенню елементів відповідного стовпця матриці A розмірністю M на N.

У даному прикладі кількість елементів вектора B дорівнює кількості стовпців матриці A. Для рішення задачі необхідно організувати перебір елементів двовимірного масиву по стовпцях. Зовнішнім повинен бути цикл по параметру ***j***, внутрішнім цикл по параметру ***i***. Це дасть можливість після завершення обробки кожного стовпця обчислювати відповідні елементи одномірного масиву B. Блок-схема алгоритму наведена на рис. 12.4.

Ввід елементів масиву A здійснюються по рядкам (блоки 2-5). У зовнішньому циклі по параметру ***j*** вибирається стовпець масиву A (блок 6), для нього обнуляється значення суми, що буде зберігається у відповідному ***j***-му елементі масиву B (блок 7). Внутрішній цикл по параметру ***i*** (блок 8) виконує перебір і підсумовування всіх елементів поточного ***j***-го стовпця масиву A (блок 9). Після завершення роботи внутрішнього циклу в ***j***-му елементі масиву B обчислюється середнє арифметичне значення елементів ***j***-го стовпця масиву A (блок 10), що потім виводиться (блок 11). На цьому закінчується тіло зовнішнього циклу й відбувається перехід на його початок, де вибирається наступний стовпець мат-

риці. Обробивши всі стовпці двовимірного масиву A, алгоритм завершить свою роботу.

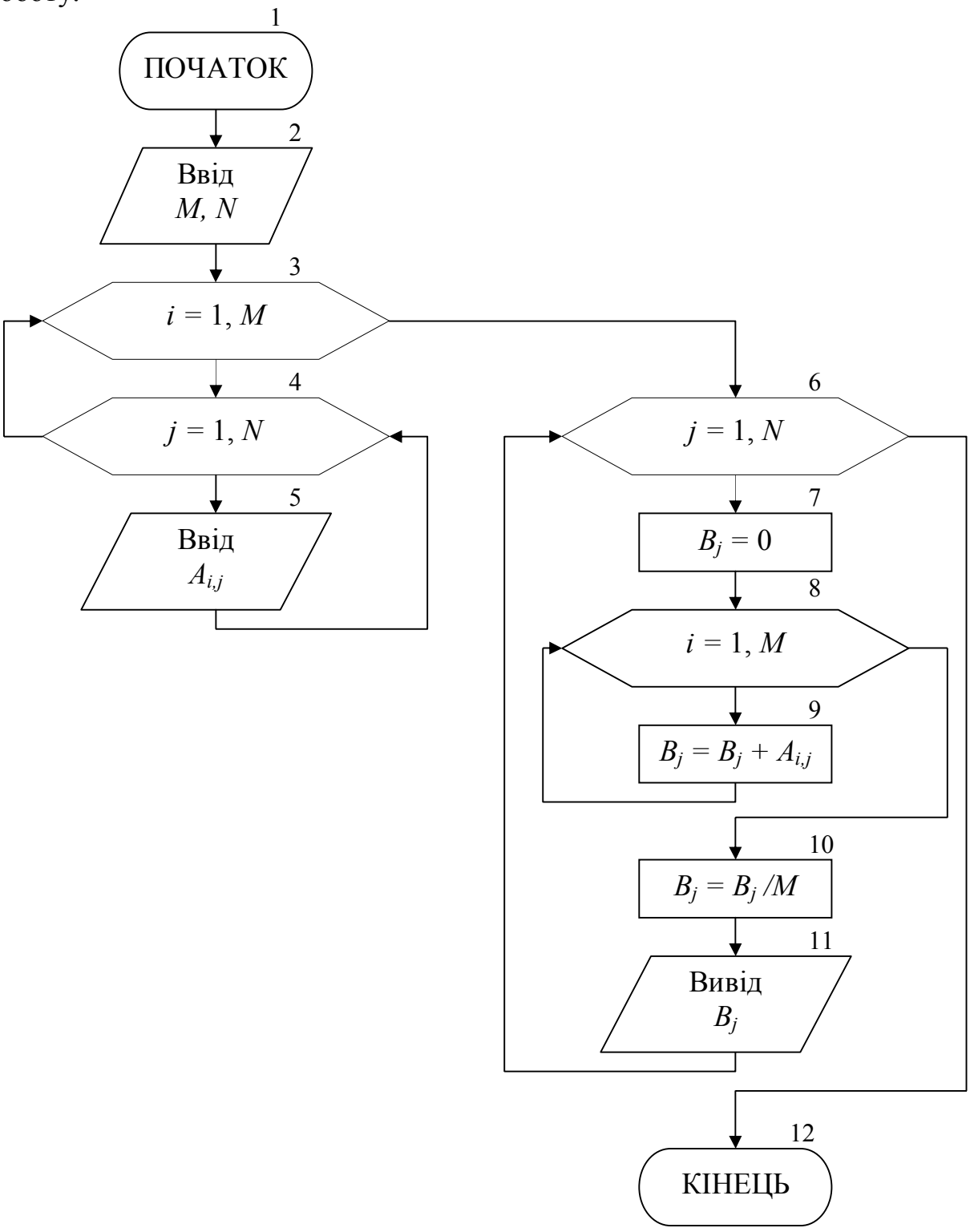



Рисунок 12.4. Блок-схема алгоритму для прикладу 12.2

## 12.2. Задачі обробки двовимірних масивів

Принципи пошуку максимального або мінімального елементів у двовимі-

рних масивах нічим не відрізняються від аналогічних принципів, які використовувані для одномірних масивів. Тільки як параметри такого елемента визначаються номери рядка й стовпця.

**Головною діагоналлю** квадратної матриці називається діагональ, що з'єднує верхній лівий кут матриці із правим нижнім кутом. Для елементів, які розташовані на головній діагоналі дотримується співвідношення між індексами: $i = j$. Для елементів, які розташовані нижче головної діагоналі: $i > j$. Для елементів, які розташовані вище головної діагоналі: $i < j$. Перебираючи елементи матриці й перевіряючи співвідношення між індексами, завжди можна визначити, де розташований поточний елемент.

Для перебору тільки елементів квадратної матриці, які розташовані вище головної діагоналі можна організувати два вкладених цикли з наступними параметрами: $i = 1 \div N\text{-}1$ і $j = i + 1 \div N$, де $N$ – розмірність матриці. Аналогічно для перебору елементів, які розташовані нижче головної діагоналі досить організувати два цикли з наступними параметрами: $i = 2 \div N$ і $j = 1 \div i - 1$.

**Побічною діагоналлю** квадратної матриці називається діагональ, що з'єднує нижній лівий кут матриці із правим верхнім кутом. Для елементів, які розташовані на побічній діагоналі дотримується співвідношення між індексами: $i = N - j + 1$. Для елементів, які розташовані нижче побічної діагоналі: $i > N - j + 1$. Для елементів, які розташовані вище – $i < N - j + 1$.

**Приклад 12.3.** У кожному рядку квадратної матриці A розмірністю N на N знайти найбільший елемент і поміняти його місцями з елементом головної діагоналі.

У даній задачі необхідно організувати порядковий перебір елементів матриці. У кожному рядку будемо шукати максимальний елемент, при цьому для однозначного його визначення досить зберігати тільки номер стовпця, у якому він знаходиться в поточному рядку. Блок-схема алгоритму наведена на рис. 12.5.

У блоках 2-5 вводиться по рядкам квадратна матриця A розмірністю N на N. Зовнішній цикл по параметру $i$ (блок 6) вибирає поточний $i$-й рядок матриці, у якому буде виконуватися пошук максимального елемента. Спочатку за максимум приймається перший елемент $i$-го рядка (блок 7). Внутрішній цикл перебирає елементи рядка, починаючи із другого (блок 8), і порівнює їх з елементом, який прийнятий за максимальний (блок 9). Якщо поточний елемент виявляється більше, ніж елемент, що прийнятий за максимум, то в змінної ***jmax*** запам'ятовується його позиція в $i$-му рядку, тобто номер стовпця, у якому цей елемент перебуває (блок 10). У противному випадку поточний елемент пропускається. Після виходу із внутрішнього циклу в елемент головної діагоналі $A_{i,i}$ поточного $i$-го рядка заноситься значення максимального елемента цього рядка $A_{i,jmax}$ (блок 11). На цьому закінчується обробка поточного рядка масиву A (тіло

зовнішнього циклу) і відбувається вибір наступного рядка масиву. Після завершення обробки всіх рядків двовимірного масиву A, у блоках 12-14 виводиться перетворена матриця й алгоритм завершує свою роботу.

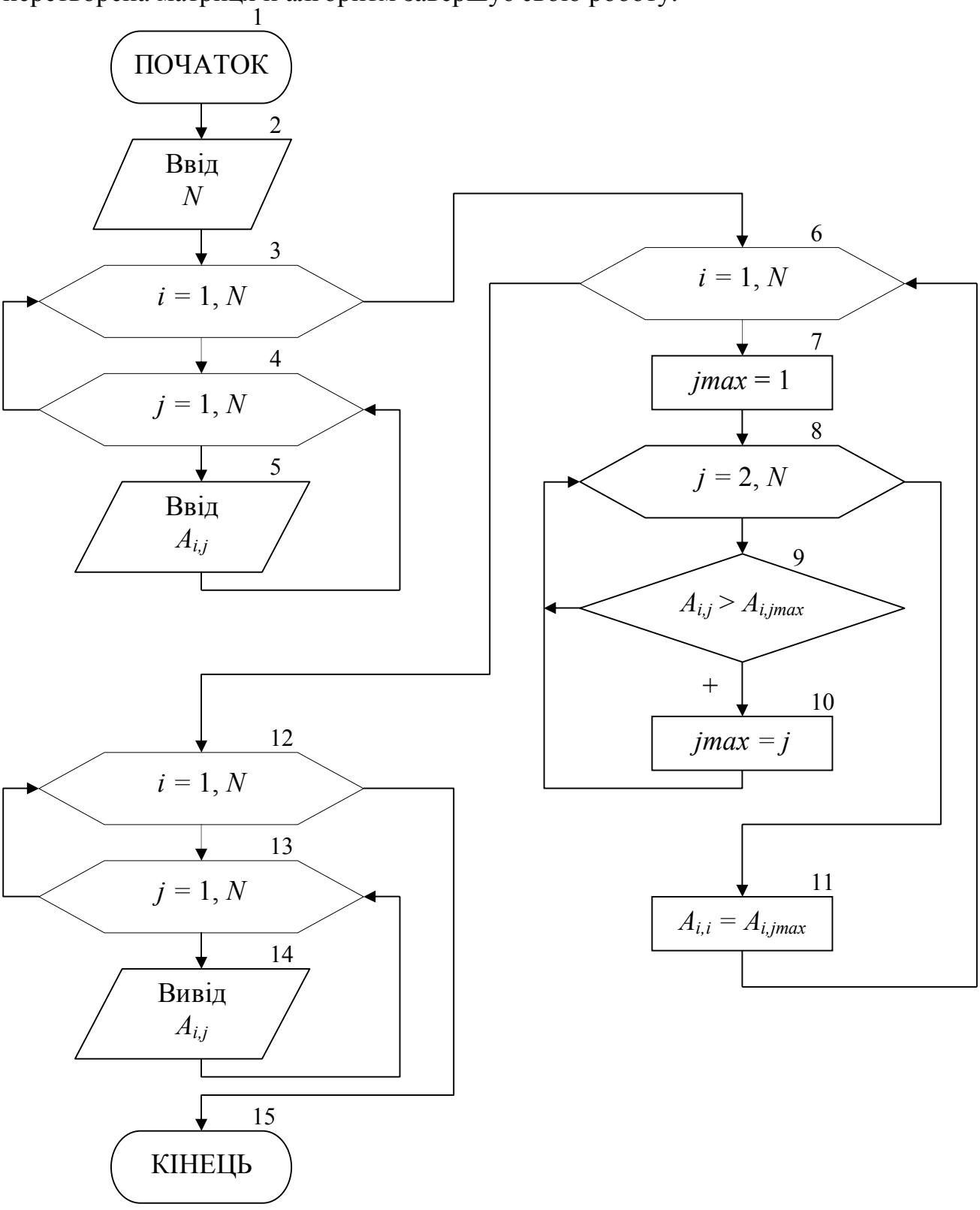



Рисунок 12.5. Блок-схема алгоритму для прикладу 12.3

При рішенні деяких задач виникає необхідність виконати множення матриці на матрицю або матриці на вектор. Розглянемо більш докладно ці операції.

Результатом множення матриці A (розмірністю *M* на *K*) на матрицю B (розмірністю *K* на *N*) буде матриця C (розмірністю *M* на *N*), тобто

$$A(\boldsymbol{M, K}) \times B(\boldsymbol{K, N}) = C(\boldsymbol{M, N}).$$

Причому елементи матриці C визначаються як сума добутків елементів відповідних рядка матриці A и стовпця матриці B:

$$C_{i,j} = \sum_{l=1}^{K} A_{i,l} \cdot B_{l,j}, \quad де \quad i = \overline{1,M}; \quad j = \overline{1,N}.$$

На рис. 12.6 наведений фрагмент блок-схеми алгоритму множення матриць A и B, що використовує три вкладених цикли (процес вводу вхідних масивів A и B не показаний).

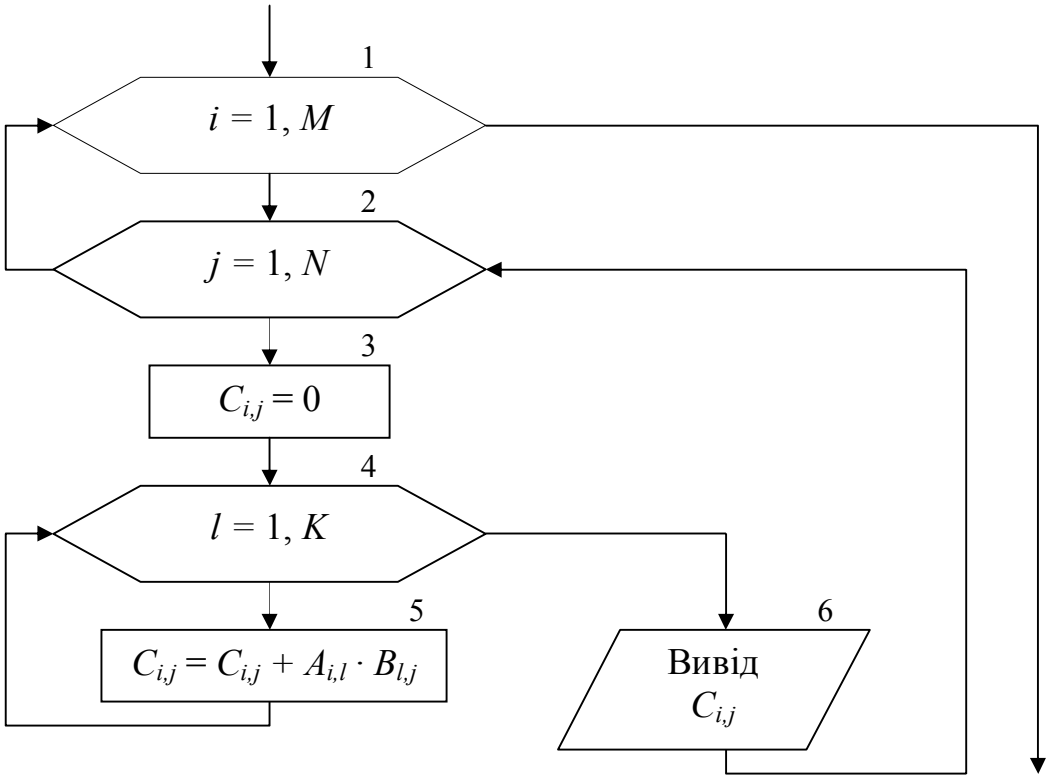



Рисунок 12.6. Блок-схема алгоритму множення матриць

Зовнішній цикл по параметру ***i*** (блок 1) вибирає рядок матриці A, перший внутрішній цикл по параметру ***j*** (блок 2) вибирає стовпець матриці B. У теж час обидва цикли використовуються для перебору елементів матриці C, кожний елемент $\boldsymbol{C_{i,j}}$ спочатку обнуляється (блок 3). Другий внутрішній цикл по параметру ***l*** (блок 4) обчислює значення поточного елемента $\boldsymbol{C_{i,j}}$, як суму добутків значень відповідних елементів ***i***-го рядка матриці A и ***j***-го стовпця матриці B (блок 5).

Результатом множення матриці A (розмірністю *M* на *N*) на вектор X (розмірністю *N*) буде вектор Y (розмірністю *M*), тобто

$$A(\boldsymbol{M, N}) \times X(\boldsymbol{N, 1}) = Y(\boldsymbol{M,1}).$$

Причому елементи вектора Y визначаються як сума добутків елементів

відповідного рядка матриці A и вектора X:

$$Y_i = \sum_{j=1}^{N} A_{ij} X_j, \quad де \quad i = \overline{1, M} .$$

На рис. 12.7 наведений фрагмент блок-схеми алгоритму множення матриці A и вектора X, що використовує два вкладених цикли (процес вводу вхідних масиву A и вектора X не показаний).

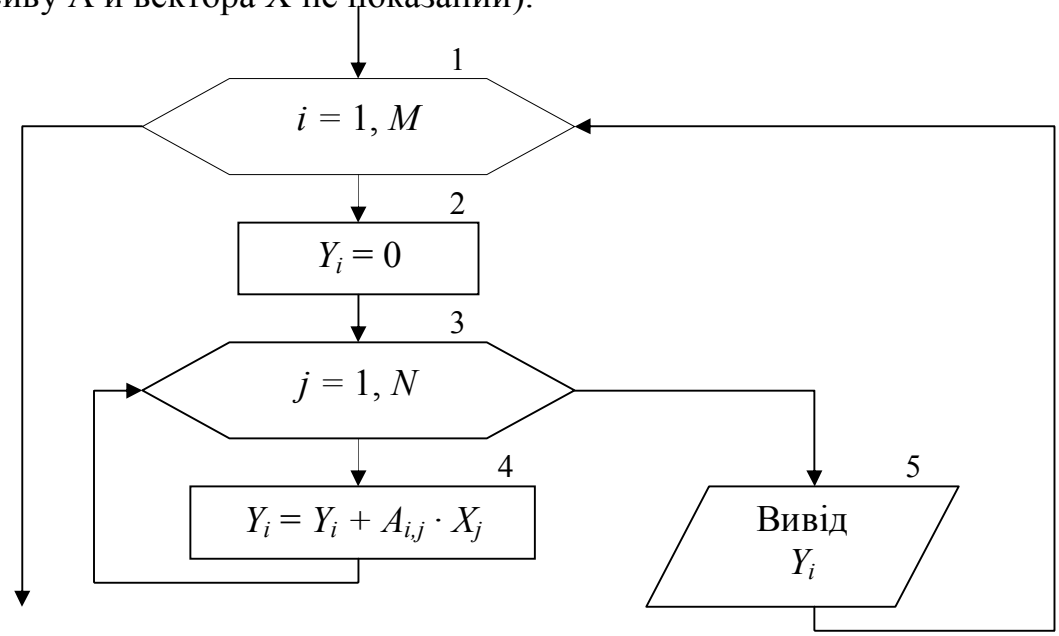



Рисунок 12.7. Блок-схема алгоритму множення матриці на вектор

Зовнішній цикл по параметру ***i*** (блок 1) вибирає рядок матриці A, і обнуляє відповідний їй елемент вектора Y (блок 2). Внутрішній цикл по параметру ***j*** (блок 3) обчислює значення поточного елемента $\boldsymbol{Y_j}$, як суму добутків значень відповідних елементів ***i***-го рядка матриці A и вектора X (блок 4).

Також з курсу вищої математики згадаємо деякі типи квадратних матриць:

- матриця називається одиничною, якщо всі елементи нулі, а на головній діагоналі одиниці;
- матриця називається діагональною, якщо всі елементи нулі, крім головної діагоналі;
- матриця нульова, якщо всі елементи нулі;
- матриця називається верхньотрикутною, якщо всі елементи нижче головної діагоналі нулі;
- матриця називається нижньотрикутною, якщо всі елементи вище головної діагоналі нулі;
- матриця називається симетричною, якщо $A=A_T$ .

Для перевірки, що матриця B є зворотною матриці A, потрібно перевірити умову A × B = E (E – одинична матриця).

**Приклад 12.4.** Перевірити, чи є задана квадратна матриця A (розмірністю *N* на *N*) одиничною (рис. 12.8).

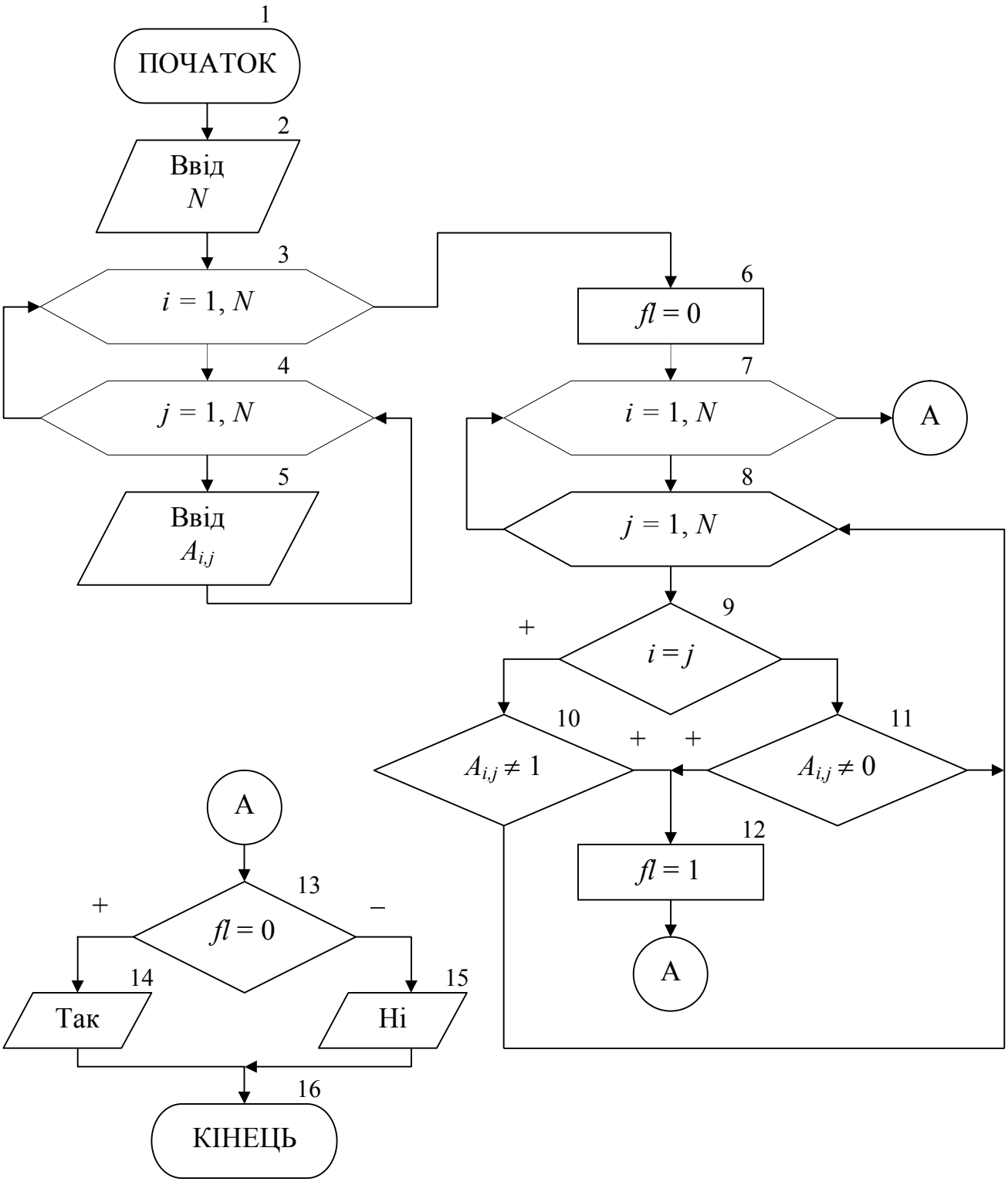



Рисунок 12.8. Блок-схема алгоритму для прикладу 12.4.

Виконавши ввід вхідної матриці A (блоки 2-5) припускаємо, що вона є одиничною, прапор *fl* встановлюється в нуль (блок 6). Потім організовуємо перебір елементів матриці A (блоки 7-8) і перевіряємо, чи розташований поточ-

ний елемент на головній діагоналі (блок 9). Якщо елемент лежить на головній діагоналі матриці й $A_{i,j} \neq 1$ (блок 10), то $fl = 1$ (блок 12), тобто матриця не одинична. Аналогічна дія виконується, якщо елемент не лежить на головній діагоналі матриці й $A_{i,j} \neq 0$ (блок 11). Після виходу із вкладених циклів (якщо $fl$ стає рівним 1, то відбувається достроковий вихід із циклів) перевіряється значення прапора $fl$ (блок 13) і виводиться повідомлення про те, чи є матриця А одиничною чи ні (блоки 14-15).

## 12.3. Приклад виконання лабораторної роботи «Обробка двовимірних масивів»

**Завдання**. Скласти блок-схему алгоритму рішення поставленого завдання обробки двовимірного масиву.

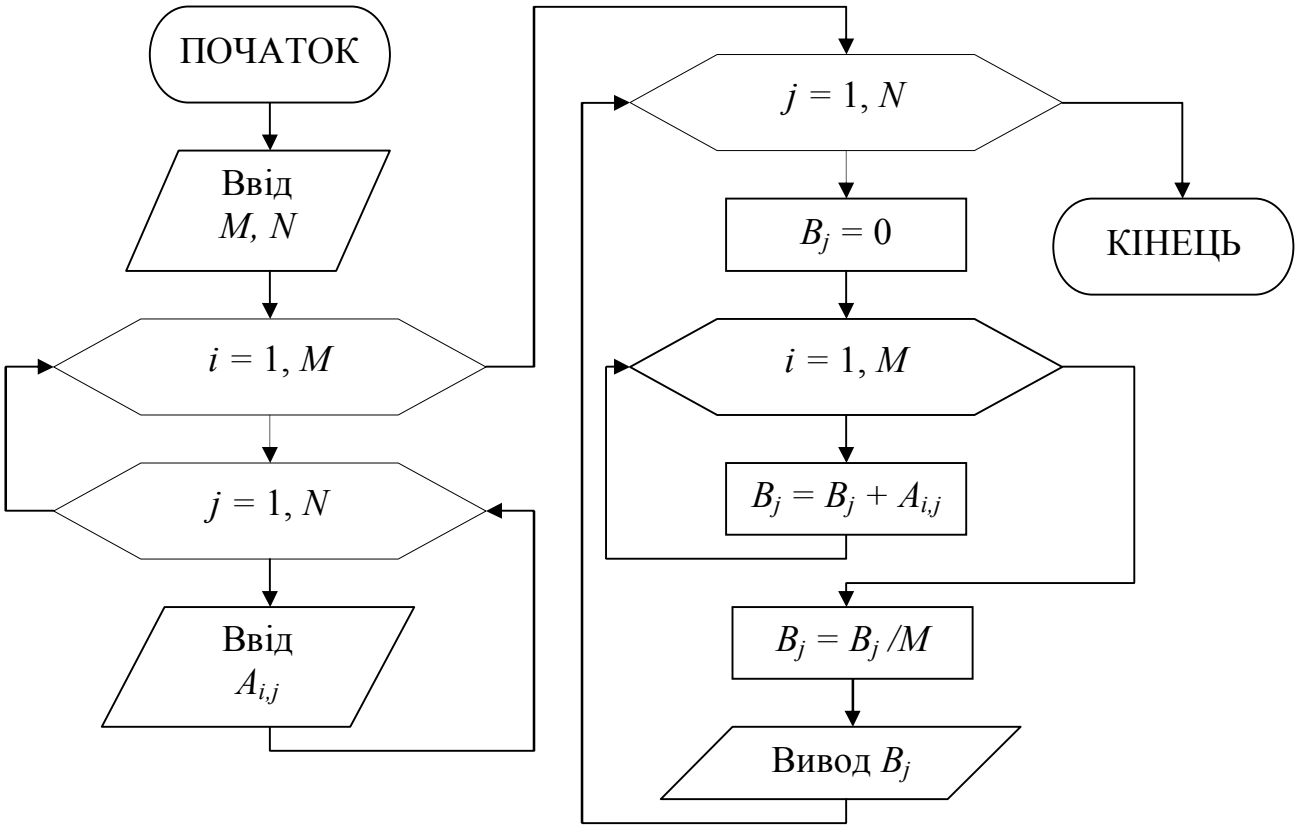
Зміст звіту по лабораторній роботі.

1. Вхідні дані: двовимірний масив $A$, розмірністю $i = 1 \div M, j = 1 \div N$.

2. Постановка задачі: сформувати одномірний масив $B = (b_1, b_2, \ldots, b_N)$, кожен елемент якого дорівнює середньому арифметичному значенню елементів відповідного стовпця двовимірного масиву $A$.

3. Вихідні дані: масив $B$.

4. Блок-схема алгоритму:



### 12.4. Завдання для самостійної роботи

**Завдання**. Скласти блок-схему алгоритму рішення поставленого завдання обробки двовимірного масиву.

1. Виконати транспонування матриці А (поміняти місцями рядки й стовпці матриці).

2. Поміняти місцями елементи головної й побічної діагоналі матриці А розмірністю *K* на *K*.

3. Перетворити вхідну матрицю А(*N*, *M*) так, щоб перший нульовий елемент кожного рядка був замінений середнім арифметичним елементів цього рядка.

4. Перетворити матрицю А(*M, N*) так, щоб рядки з непарними індексами були впорядковані по убуванню, с парними – по зростанню.

5. Задано матрицю А(*M, N*). Сформувати вектор Р(*M*), у який записати номера рядків максимальних елементів кожного стовпця.

6. Визначити середнє арифметичне додатних елементів матриці А(*K*, *K*), які розташовані поза діагоналями матриці. Якщо немає додатних елементів, то поміняти місцями елементи головної й побічної діагоналей.

7. У кожному рядку матриці F(*k,k*) елемент, що лежить на головній діагоналі, якщо це від’ємне число, замінити сумою раніше розташованих елементів.

8. У кожному стовпці матриці А(*n,m*) мінімальний елемент замінити сумою позитивних елементів цього ж стовпця.

9. Перевірити, чи є матриця А(*n,n*) діагональною (всі елементи нулі, крім головної діагоналі), одиничною (всі елементи нулі, на головній діагоналі тільки одиниці) або нульовою (всі елементи нулі).

10. З матриці А(*K,L*) сформувати вектор В(*K*), кожний елемент якого дорівнює кількості додатних елементів відповідного рядка матриці А, і вектор С(*K*), кожний елемент якого дорівнює кількості від’ємних елементів відповідного рядка матриці А.

## Лекція №13. VISUAL BASIC FOR APPLICATION (VBA)

### 13.1. Загальні зведення про об'єктно-орієнтоване програмування

***Об'єктне-орієнтоване програмування*** (ООП) являє собою методику аналізу, проектування й написання додатків за допомогою об'єктів - фрагментів коду, що володіють властивостями й методами. Об'єкти створюються на базі класів і можуть моделювати правила обробки даних, різні ситуації й фізичні предмети.

Самим фундаментальним об'єктом **VISUAL BASIC** є об'єкт ***форма***, завдяки якої додатки є візуальними. Форми мають множину властивостей, які є відкритими й можуть вільно змінюватися програмами. Форми мають також множину методів і подій.

Таким чином, ООП дає можливості користувачу, що створює додаток, розподілити функції програми по декількох незалежних об'єктах. Оптимізація об'єктів знижує ризик небажаної взаємодії фрагментів програми.

***Властивості*** описують об'єкт, кожний створюваний об'єкт має ім'я класу, по якому його можна відрізнити від інших об'єктів.

***Методами*** називаються дії, які виконуються об'єктом. Наприклад, об'єкт - вікно може відображати або приховувати себе на екрані.

***Спадкуванням*** називається здатність об'єкта зберігати атрибути класу – батька. Щоб створити екземпляр об'єкта в VISUAL BASIC, варто вказати на його приналежність деякому класу й потім скористатися ключовим словом New. Новий об'єкт успадковує властивості й методи батька.

***Інкапсуляцією*** називається механізм, завдяки якому дані й методи об'єкта приховуються від зовнішнього миру. Інкапсуляція запобігає сторонньому доступу до конфіденційної інформації. В VISUAL BASIC інкапсуляцію забезпечує ключове слово Private.

***Поліморфізм*** – здатність об'єкта приймати різні форми. Поліморфізм дозволяє додавати, видозмінювати або видаляти деякі особливості поводження довільного об'єкта.

## 13.2. Інтегроване середовище розробки додатків мовою VBA

Для виклику інтегрованої у MS Excel середи розробки додатків (IDE) необхідно виконати команду: **Сервис – Макрос – Редактор Visual Basic.** Вікно редактору VBA (рис. 13.1) складається з декількох компонентів: головного меню, панелі інструментів, вікна проекту, вікна властивостей, панелі елементів, конструктора форм, вікна контрольних значень і декількох інших допоміжних вікон.

***Головне меню*** Visual Basic складається з декількох пунктів.

Меню **Файл (File)** призначений для роботи з файлами, з яких утворяться додатки. У ньому можна створювати, зберігати й друкувати проекти.

Меню **Правка (Edit)** виконує стандартні операції з буфером обміну – вирізання, копіювання й вставка. Вони застосовуються не тільки до фрагментів програми, але й до керуючих елементів.

У меню **Вид (View)** включаються режими перегляду різних компонентів і інструментів. Переглядати можна форми й програмні модулі.

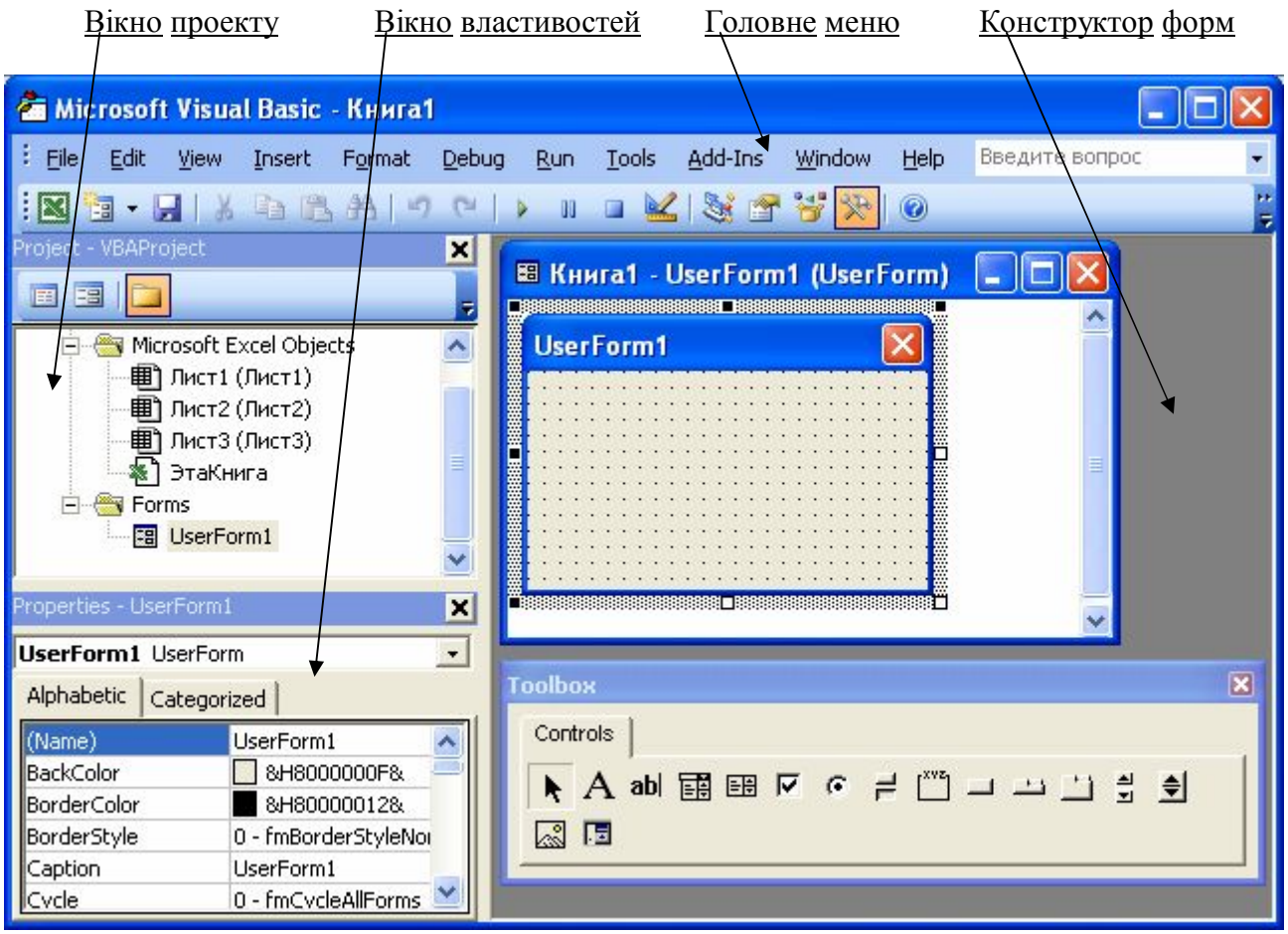



Рисунок 13.1. Вікно редактору VBA

Меню **Вставка (Insert)** дозволяє додавати процедури, форми, модулі й модулі класу.

Команди меню **Формат (Format)** визначають розташування й розміри елементів і форм.

За допомогою команд **Меню Налагодження (Debug)** можна запустити й зупинити додаток, розставити точки переривання й вибрати для перегляду об'єкти, а також виконати інші операції, що допомагають стежити за роботою додатка.

Команди меню **Запуск (Run)** запускають і зупиняють додаток, переривають і відновляють виконання програми, що особливо зручно в процесі налагодження.

Меню **Сервис (Tools)** дозволяє включити додаткові елементи, запустити макроси й настроїти параметри редактора.

Меню **Вікно (Window)** дозволяє розташувати вікна IDE (каскадне або мозаїчне розташування), упорядкувати значки згорнутих форм, а також створює список, що дозволяє швидко перейти до одного з відкритих вікон IDE.

Меню **Допомога (Help)** – допомога користувачу.

Для швидкого виклику головного меню необхідно натиснути F10.

Панель інструментів (**Toolbox**) знаходиться під головним меню. Якщо

вона відсутня, необхідно виконати команду: **Вид – Панели инструментов – Standart.**

**Вікно проекту (Project)** нагадує собою вікно провідника Windows, і призначено для швидкого перегляду складових проекту, що об'єднує в собі всі об'єкти, які складають додаток.

**Вікно властивостей (Properties)** відображає різні атрибути виділеного об'єкта. Всі об'єкти (форми, елементи, що управляють й тобто) мають атрибути, які змінюють не тільки зовнішній вигляд об'єкта, але і його поведінку. Всі ці атрибути називаються властивостями. Отже, кожний об'єкт має набір властивостей.

**Конструктор форм** розташований у центрі екрана редактора VBA. Тут виводиться або зображення форми, що дозволяє робити візуальне конструювання макета форми й елементів, які розташовані на ній, або вікно програми.

## 13.3. Особливості програмування мовою VBA

Процес розробки програм мовою VBA складається з декількох етапів, залежно від кінцевого результату. Якщо необхідно одержати програму, яка буде робити визначені обчислення або дії, то досить створити програмний модуль, що містить текст вхідного коду. Для запуску цієї програми можна створити кнопку, при натисканні якої програма буде виконуватися. Для цього в MS Excel необхідно включити панель інструментів за допомогою команди: **Вид – Панели инструментов – Элементы управления**, а потім створити кнопку. Або виконати програму за допомогою команди: **Сервис – Макрос – Макросы.**

Розробка "повноцінної" програми буде включати два етапи.

***Перший етап*** – етап візуального програмування, на якому створюється вікно (форма) програми, де розташовуються необхідні елементи управління.

***Другий етап*** – етап програмування у вхідному коді, на якому створюються частини програми, що виконуються у відповідь на певні події. Подією є, наприклад, щиглик лівою кнопкою миші на командній кнопці (подія Click), натискання клавіші на клавіатурі (подія KeyPress) і тобто.

## 13.4. Процедури й функції

Основу програмного коду в VBA складають процедури й функції, які записуються в модулі. ***Модуль*** (**Module**) являє собою текстовий ASCII-файл із програмним кодом, що містить підпрограми, змінні й константи.

Проект може складатися із безлічі програмних модулів. Для їхнього створення необхідно виконати команду **Вставка – Модуль.**

Загальні принципи організації програм VBA у модулі наступні. Звичайно текст програми починається з опцій, які керують описом змінних, способом по-

рівняння рядків і т.д. Потім слідує оголошення ***глобальних*** для даного модуля змінних і констант, тобто таких, які використовуються у всіх процедурах модуля. Далі розташовують безпосередньо текст функцій і процедур, що складають саму програму. Роздільником операторів в одному рядку при записі програми є символ " : ". Для перенесення оператора на інший рядок використовується символ "_" (знак підкреслення).

Іноді усередині програмного коду зручно поміщати ***коментарі*** – пояснювальний текст, що ігнорується компілятором і може бути записаний у будь-якому місці програми. Коментарі зручно також використовувати при налагодженні програми для тимчасового відключення операторів. Кожен рядок коментарів починається зі знака апострофа.

**Процедура Sub** – це відособлена сукупність операторів VBA, що виконує визначені дії. Процедура приймає деякі параметри (змінні, які передаються процедурі в якості вхідних даних), виконує програмний код і може повертати результуючі значення, які присвоюються параметрам усередині процедури. Вкладеність процедур в інші процедури не допускається.

Структура процедури наступна:

*[ДОСТУП]* **Sub** *ІМ'Я_ ПРОЦЕДУРИ([СПИСОК_ПАРАМЕТРІВ])*
*ТІЛО_ПРОЦЕДУРИ*
**End Sub**

Ключове слово *ДОСТУП* є необов'язковим і визначає область видимості процедури. **Public** вказує, що процедура доступна для всіх інших процедур у всіх модулях (глобальна). **Private** вказує, що процедура доступна для інших процедур тільки того модуля, у якому вона описана (локальна). *СПИСОК_ПАРАМЕТРІВ* також є необов'язковим елементом і дозволяє передавати процедурі різні вихідні дані при виклику, які називаються формальними параметрами. При цьому ключове слово **Dim** не вказується.

*ТІЛО_ПРОЦЕДУРИ* складається з описової частини й блоку операторів, що виконуються один за іншим. Якщо необхідно припинити виконання процедури в деякому конкретному місці, це можна зробити за допомогою оператора **Exit Sub**.

*ІМ'Я ПРОЦЕДУРИ* – це будь-який ідентифікатор, визначений користувачем. **Ідентифікатор** – це послідовність літер, цифр і символу підкреслення, що починається з літери (пробіли усередині ідентифікатора неприпустимі). Ім'я процедури завжди визначається на рівні модуля. Для використання процедури в тексті програми (тобто для її виклику), необхідно вказати ім'я процедури й список фактичних параметрів, які повинні по типу й порядку проходження збігатися з формальними параметрами.

**Функція Function** багато в чому схожа на процедуру, але на відміну від неї при виклику завжди повертає значення. Функція отримує параметри, яки називані **аргументами**, і виконує з ними деякі дії, результат яких повертається

функцією. Структура функції наступна:

*[ДОСТУП]* **Function** *ІМ'Я_ФУНКЦІЇ(СПИСОК_АРГУМЕНТІВ)* **As** *ТИП*
*ТІЛО_ФУНКЦІЇ*
*ІМ'Я_ФУНКЦІЇ=ВИРАЗ*
**End Function**

*ТИП* визначає тип даних результату, який повертається функцією. У тілі функції обов'язково повинен бути присутнім, принаймні, один оператор, що присвоює імені функції значення виразу, який обчислюється. Дострокове завершення функції можливо за допомогою оператора **Exit Function.** У програмі виклик функції здійснюється за допомогою оператора присвоювання, у правій частині якого вказується ім'я функції з переліком фактичних параметрів, як і будь-якої іншої убудованої функції, наприклад, Sqr, Cos або Chr.

Процедури й функції, які не описані явно за допомогою ключових слів **Public** або **Private**, за замовчуванням є загальними.

Для швидкого додавання в модуль підпрограм зручно скористатися командою **Вставка – Процедура**. У вікні, що з'явилося, потрібно вибрати необхідні опції.

В MS Excel з функціями, які створені користувачем, можна працювати за допомогою ***Майстра функцій*** точно так само, як і з вбудованими функціями.

## 13.5. Стандартні типи даних, опис змінних

Всі значення, з якими працює програма, зберігаються в змінних, константах, масивах. **Змінною** називається область пам'яті, у якій можуть зберігатися різні значення. Кожна змінна в VBA має свій тип, який вказує, що може зберігати змінна: ціле число, рядок, дату й тобто.

У процесі виконання програми значення змінних можуть змінюватися за допомогою оператора присвоювання або вводу. Таким чином, розрізняють:

***опис змінних*** (визначення їхнього типу);

***ініціалізацію змінних*** (присвоєння ним початкових значень).

Змінну в VBA можна описати за допомогою наступної конструкції:

**Dim** *ІМ'Я_ЗМІННОЇ* **As** *ТИП_ЗМІННОЇ*

*ТИП_ЗМІННОЇ* – це один зі стандартних типів даних VBA (табл. 13.1). При цьому в одному рядку можна описувати декілька змінних через кому.

Наприклад:

Dim A As Integer– оголошується змінна A цілого типу;

Dim E As Single, D As Single – оголошуються змінні E, D дійсного типу;

VBA дозволяє не описувати змінні перед використанням у програмі. У цьому випадку за замовчуванням використовується тип Variant. Змінні цього типу можуть зберігає все, що в них помістять, тобто їхній тип змінюється зале-

жно від останнього присвоєння. Однак рекомендується всі змінні, які будуть використовуватися в програмі, описувати явно. Для заборони використання змінних, які не були явно описані на початку програми, необхідно вставити рядок **Option Explicit**.

Таблиця 13.1. Стандартні типи даних VBA

| Типи даних | Розмір пам'яті (байт) | Призначення |
|---|---|---|
| **Boolean** (Логічний) | 2 | Містить значення True (Істина) або False (Неправда). |
| **Byte** (Позитивне ціле) | 1 | Містить цілі позитивні числа в діапазоні від 0 до 255. |
| **Integer** (Ціле) | 2 | Містить цілі числа в діапазоні від –32768 до 32768 |
| **Long** (Довге ціле) | 4 | Містить цілі числа в діапазоні від –2147483648 до 2147483647 |
| **Single** (Число із плаваючою точкою ) | 4 | Містить речовинні числа по абсолютній величині від 1,401298E-45 до 3,402823E+38 |
| **Date** (Дата) | 8 | Містить дати від 1.01.0100г. до 1.12.9999р. |
| **Double** (Число із плаваючою точкою подвійної точності) | 8 | Містить речовинні числа по абсолютній величині від 4,940656438E-324 до 1,79769313E+308 |
| **Object** (Об'єкт) | 4 | Містить посилання на будь-який заданий об'єкт. |
| **String** (Строковий) | 10 + довжина | Містить текстові або строкові значення довжиною від 0 до $2 \cdot 10^9$. |
| **Currency** (Грошовий) | 8 | Містить грошові величини в діапазоні від –922337203685477,6 до 922337203685477,6 |
| **Variant** (Варіант) | - | Універсальний тип загального призначення, що може зберігати значення більшості інших типів. |

Часто при написанні програм необхідно використовувати ті самі постійні значення: числа, рядка, дати й тобто. У цьому випадку можна задати константу за допомогою однієї з наступних конструкцій:

**Const** ІМ'Я_ПОСТІЙНОЇ = ВИРАЗ

або

**Const** ІМ'Я_ПОСТІЙНОЇ **As** ТИП_ПОСТІЙНОЇ = ВИРАЗ

Наприклад: Const FileName = “lab.xls”

Const Pi As Double = 3.14159

## 13.6. Запис виразів

Ініціалізація змінних виконується за допомогою оператору **присвоювання**, який має наступний синтаксис:

*ІМ'Я_ ЗМІННОЇ = ВИРАЗ*

*ВИРАЗ* – це будь-які арифметичні (табл. 13.2) й логічні оператори, оператори порівняння й конкатенації, які написані за правилами VBA. Математичний вираз завжди записується в один рядок. Результат обчислення виразу заноситься в зазначену змінну. Виконувати присвоювання можна тільки між сумісними змінними.

Таблиця 13.2. Арифметичні оператори

| Оператор | Опис |
|---|---|
| Оператор ^ | Зводить число в ступінь. |
| Оператор * | Повертає добуток двох чисел. |
| Оператор / | Повертає результат ділення двох чисел. |
| Оператор \ | Повертає результат цілого ділення двох чисел. |
| Оператор Mod | Повертає остача при цілому діленні двох чисел. |
| Оператор + | Повертає суму двох чисел. |
| Оператор – | Повертає різниця двох чисел, або змінює знак. |

Якщо вираз містить декілька операторів, то значення компонентів виразу розраховуються в визначеному порядку, що називають пріоритетом операторів.

Якщо вираз містить оператори різних типів, то першими виконуються арифметичні операції, слідом за ними операції порівняння, а останніми логічні операції. Всі оператори порівняння мають рівний пріоритет, тобто виконуються в порядку їхнього розташування у виразі ліворуч праворуч. Арифметичні й логічні оператори виконуються в порядку їхнього пріоритету (табл. 13.3).

Розташовані поруч у виразі оператори множення й ділення виконуються ліворуч праворуч. У такому ж порядку виконуються розташовані поруч оператори додавання й вирахування. Оператори усередині круглих скобок завжди виконуються раніше, ніж оператори поза скобками. Порядок виконання операторів, що стоять усередині скобок, обумовлюється старшинством операторів.

Таблиця 13.3. Пріоритет операторів

| Арифметичні | Порівняння | Логічні |
|---|---|---|
| Піднесення в ступінь (^) | Дорівнює (=) | Not |
| Зміна знака (-) | Не дорівнює (<>) | And |
| Множення й ділення (*,/). | Менше (<) | Or |
| Ціле ділення (\). | Більше ( >) | Xor |
| Ділення по модулю (Mod). | Менше або дорівнює (<=) | |
| Додавання й вирахування (+,-). | Більше або дорівнює (>=) | |

Наприклад, математичний вираз:

$$y = \frac{x^2 - 3 \cdot x^{1/3}}{(0.2 \cdot x + 1) \cdot (x - 4)}$$

на VBA буде мати такий вигляд:

```
y = (x ^ 2 – 3 * x ^ (1/3))/((0.2 * x + 1) * (x – 4))
```

У VBA використовуються наступні стандартні функції (табл. 13.4).

Таблиця 13.4. Стандартні функції VBA

| **Функція** | **Опис** |
|---|---|
| CDate(вираз) | Перетворить вираз в тип Date |
| CInt(вираз) | Перетворить вираз в тип Integer |
| CLng(вираз) | Перетворить вираз в тип Long |
| CSng(вираз) | Перетворить вираз в тип Single |
| CStr(вираз) | Перетворить вираз в тип String |
| abs(аргумент) | Повертає значення, тип якого збігає з типом переданого аргументу, яке рівне абсолютному значенню указаного числа. |
| atn(аргумент) | Повертає значення типу Double, що містить арктангенс числа. |
| cos(аргумент) | Повертає значення типу Double, що містить косинус кута. |
| int(аргумент) | Повертає значення типу, що збігає з типом аргументу, яке містить цілу частину числа. |
| log(аргумент) | Повертає значення типу Double, що містить натуральний логарифм числа. |
| exp(аргумент) | Повертає значення типу Double, що містить результат піднесення числа $e$ в указаний ступінь. |
| sin(аргумент) | Повертає значення типу Double, що містить синус кута. |
| sqr(аргумент) | Повертає значення типу Double, що містить квадратний корінь числа. |
| tan(аргумент) | Повертає значення типу Double, що містить тангенс кута. |

Наприклад, математичний вираз:

$$y = \frac{\sqrt{x - 3} + 1}{\sin(2 \cdot x + 1) + |x - 4|}$$

на VBA буде мати наступний вигляд:

```
y = (Sqr(x – 3) + 1)/(Sin(2 * x + 1) + Abs(x – 4))
```

# Лекція №14. ОБ'ЄКТИ, ВЛАСТИВОСТІ Й МЕТОДИ VBA. ОПЕРАТОРИ ВВОДУ-ВИВОДУ. УМОВНИЙ ОПЕРАТОР

## 14.1. Об'єкти, властивості й методи VBA

Одним з основних понять VBA є об'єкт. ***Об'єкт*** – це те, чим Ви управляєте за допомогою програми мовою VBA, наприклад, форма, кнопка, робочий лист і діапазон комірок MS Excel. Кожен об'єкт має деякі ***властивості.*** Наприклад, форма може бути видимою або невидимою в цей момент на екрані. Інший приклад властивості об'єкта – шрифт для відображення інформації в комірці (об'єкті) робочого аркуша.

Об'єкт містить також список методів, які можуть бути до нього застосовні. **Методи** – це те, що ви можете робити з об'єктом. Наприклад, показати форму на екрані або забрати її можна за допомогою методів Show і Hide.

Таким чином, **об'єкт** – це програмний елемент, який має своє відображення на екрані, містить деякі змінні, що визначають його властивості, і деякі методи для управління об'єктом. Наприклад, в MS Excel є багато вбудованих об'єктів.

**Range**("*ДІАПАЗОН КОМІРОК*") – задає адресу діапазону комірок (може включати тільки одну комірку).

**Cells(i, j)** – комірка, що перебуває на перетинанні і-го рядка й j-го стовпця робочого листа MS Excel (i, j – цілі числа).

**Sheets**("*ІМ'Я*") – лист.

**WorkSheets** – робочий лист.

Установка значень властивостей – це один зі способів управління об'єктами.

Синтаксис установки значення властивості об'єкта наступний:

*ОБ'ЄКТ.ВЛАСТИВІСТЬ = ВИРАЗ*

Основною властивістю об'єктів Cells і Range, є Value (значення), яке, однак, можна не вказувати. Наприклад:

Range("A5:A10").Value = 0 – у діапазон комірок A5:A10 заноситься значення 0.

Cells(2,4).Value = n – в комірку, що перебуває на перетинанні 2-го рядка й 4-го стовпця (комірка з адресою "D2"), заноситься значення змінної n.

Синтаксис читання властивостей об'єкта наступний:

*ЗМІННА = ОБ'ЄКТ.ВЛАСТИВІСТЬ*

Наприклад:

xn = Cells(1,2).Value або xn = Range("B1").Value – змінної xn присвоюється значення з комірці B1 поточного робочого листа.

Синтаксис застосування методів до об'єкта:

*ОБ'ЄКТ.МЕТОД*

Наприклад: Range("A5:A10").Clear – очищається діапазон комірок A5:A10.

Range("A2:B10").Select – вибирається діапазон комірок A2:B10.

В MS Excel є багато об'єктів, причому деякі з них містять інші об'єкти. Наприклад, робоча книга містить робочі листи, робочий лист містить діапазон комірок і тобто. Об'єктом найвищого рівня є **Application** (додаток). Якщо ви змінюєте його властивості або викликаєте його методи, то результат застосовується до поточної роботи MS Excel. Наприклад:

Application.Quit – завершення роботи **з** Excel.

Відзначимо, що крапка після ім'я об'єкта може використовуватися для переходу від одного об'єкта до іншого. Наприклад:

Application.Workbooks("Звіт").Worksheets("Травень").Rows(2).Delete

Очищає другий рядок робочого аркуша „Травень" у робочій книзі „Звіт" .

Потрібно відзначити наступне:

• Можна не писати ім'я об'єкта Application, тому що це мається на увазі за замовчуванням.

•При роботі з підоб'єктом уже активізованого об'єкта немає необхідності вказувати утримуючий його об'єкт.

• VBA використовує деякі властивості й методи, які повертають об'єкт до якого вони відносяться (це дозволяє швидко вказувати потрібний об'єкт). Приклади таких властивостей: ActiveCell (активна комірка), ActiveSheets (активний аркуш), ActiveWorkBook (активна робоча книга). Так, установити значення активної комірки можна в такий спосіб:

ActiveCells.Value="Так".

## 14.2. Оператори вводу-виводу

Функція I**nputBox** здійснює ввід значень за допомогою вікна введення й має наступний синтаксис:

*ІМ'Я_ЗМІННОЇ* = **InputBox**(*ПОВІДОМЛЕННЯ[,ЗАГОЛОВОК]*)

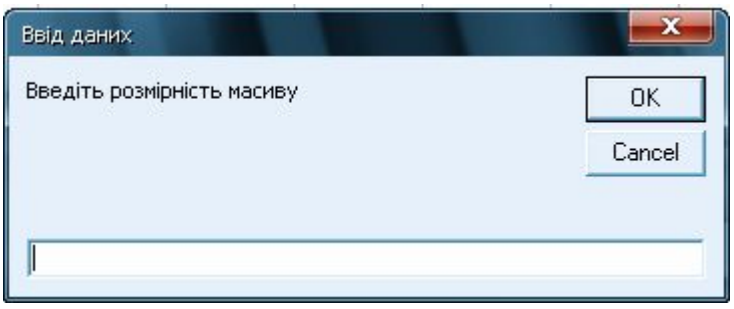
Аргументи:

*ПОВІДОМЛЕННЯ* – обов'язковий аргумент. Задає у вікні інформаційне повідомлення, яке звичайно пояснює значення величини, що вводиться.

*ЗАГОЛОВОК* – задає заголовок вікна.

Наприклад:

n = InputBox ("Введіть розмірність масиву", " Ввід даних " )



Функція **MsgBox** здійснює вивід інформації в діалоговому вікні й установлює режим очікування натискання кнопки користувачем. Вона має наступний синтаксис:

**MsgBox** *ПОВІДОМЛЕННЯ [,КНОПКИ] [,ЗАГОЛОВОК]*

Аргументи:

*ПОВІДОМЛЕННЯ* – обов'язковий аргумент, що задає у вікні виведене інформаційне повідомлення. Може складатися з декількох текстових рядків, об'єднаних знаком &. Використання в цьому аргументі Chr(13) приводить до переходу на новий рядок при виводі інформації.

*КНОПКИ* – значення цього аргументу визначає категорії кнопок, що з'являються у вікні (табл. 14.1). Від значення аргументу кнопки залежить також, чи з'являється у вікні який-небудь значок. Якщо не зазначено, які кнопки необхідно відображати у вікні повідомлень, то використовується значення за замовчуванням, що відповідає кнопці ОК.

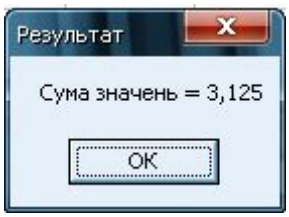
*ЗАГОЛОВОК* – задає заголовок вікна.

Функція **MsgBox** повертає значення типу Integer, що вказує, яка кнопка була натиснута в діалоговому вікні.

Наприклад:

MsgBox "Сума значень =" & CStr(S), vbOKOnly, "Результати"

виводить в окреме вікно з іменем "Результати", яке містить тільки кнопку "ОК", значення змінної S, з відповідним поясненням.




Таблиця 14.1. Припустимі значення змінної кнопки.

| Відображення | Аргумент |
|---|---|
| Кнопка ОК | VbOKOnly |
| Кнопки ОК і Скасування | VbOKCancel |
| Кнопки Та й Немає | VbYesNo |
| Кнопки Так, Немає й Скасування | VbYesNoCancel |
| Кнопки Припинити, Повторити й Ігнорувати | VbAbortRetryIgnore |
| Кнопки Повторити й Скасування | VbRetryCancel |
| Інформаційний знак | VbInformation |
| Знак питання | VbQuestion |
| Знак вигуку | VbExclamation |

## 14.3. Умовний оператор If

Оператор **If ... Then ... Else** являє собою найпростішу форму перевірки умов в VBA, і має наступний синтаксис:

**If** *УМОВА* **Then** *ОПЕРАТОР_1* **Else** *ОПЕРАТОР_2*

*УМОВА* – це вираз логічного типу. Результат виразу завжди має булевий

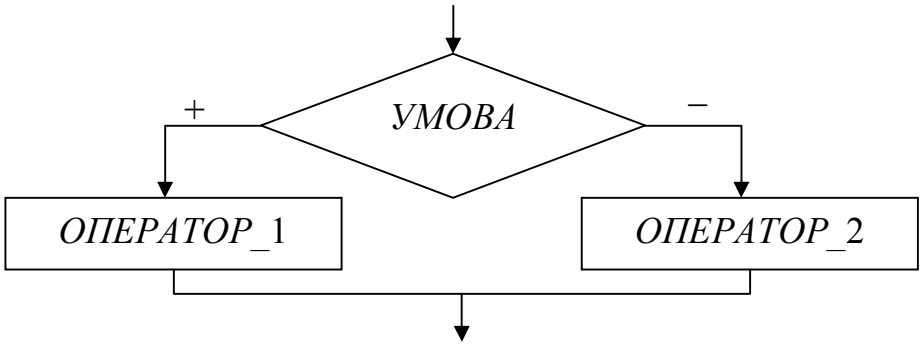
тип. Вираз може бути простим і складним. При записі простих умов можуть використовуватися всі можливі операції відносини, указані в табл. 14.2.

Таблиця 14.2. Логічні відносини.

| Операція | Назва | Вираз | Результат |
|---|---|---|---|
| = | Дорівнює | A = B | True, якщо A дорівнює B |
| <> | Не дорівнює | A<>B | True, якщо A не дорівнює B |
| > | Більше | A > B | True, якщо A ,більше B |
| < | Менше | A < B | True, якщо A менше B |
| >= | Більше або дорівнює | A>=B | True, якщо A більше або дорівнює B |
| <= | Менше або дорівнює | A<=B | True, якщо A менше або дорівнює B |

*ОПЕРАТОР_*1 виконується, якщо результат перевірки *УМОВИ* є **ІСТИНОЮ**, у противному випадку виконується *ОПЕРАТОР_*2.

У блок-схемі алгоритму умовному оператору відповідає наступна конструкція, при цьому оператор If ... Then ... Else записують в один рядок.




Складні умови утворяться із простих шляхом застосування логічних операцій і круглих скобок. Список логічних операцій наведений у табл. 14.3.

Таблиця 14.3. Логічні операції.

| Операція | Назва | Вираз | A | B | Результат |
|---|---|---|---|---|---|
| **Not** | Логічне заперечення | Not A | False<br>True | | True<br>False |
| **And** | Логічне И | A And B | True<br>True<br>False<br>False | True<br>False<br>True<br>False | True<br>False<br>False<br>False |
| **Or** | Логічне АБО | A Or B | True<br>True<br>False<br>False | True<br>False<br>True<br>False | True<br>True<br>True<br>False |

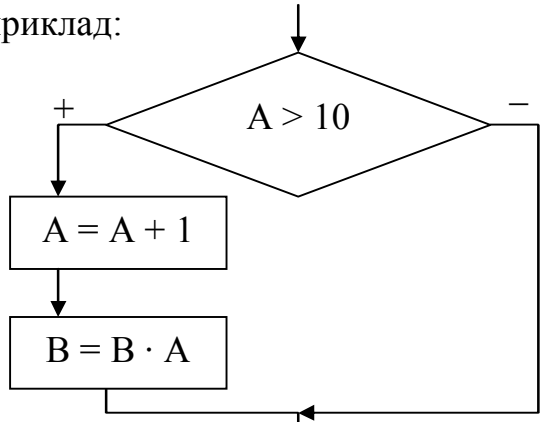
Наприклад, нерівність $3.4 \le X \le 5.2$ можна перевірити наступним логіч-

ним виразом:

X >= 3.4 And X <= 5.2

Гілка Else в умовному операторі є необов'язковою. У цьому випадку після Then можна розмістити декілька операторів, для того, щоб всі вони виконувалися, якщо умова істинно. У цьому випадку вони повинні розташовуватися в один рядок і бути розділені двокрапкою, наприклад:

If A > 10 Then A = A + 1: B = B * A




В умовному операторі припустиме використання блоку операторів замість кожного з операторів.

У цьому випадку умовний оператор має вигляд:

**If** *УМОВА* **Then**
*БЛОК_ОПЕРАТОРІВ*_1
**Else**
*БЛОК_ОПЕРАТОРІВ*_2
**End If**

В умовному операторі може перевірятися кілька умов. У цьому випадку умовний оператор може мати вигляд:

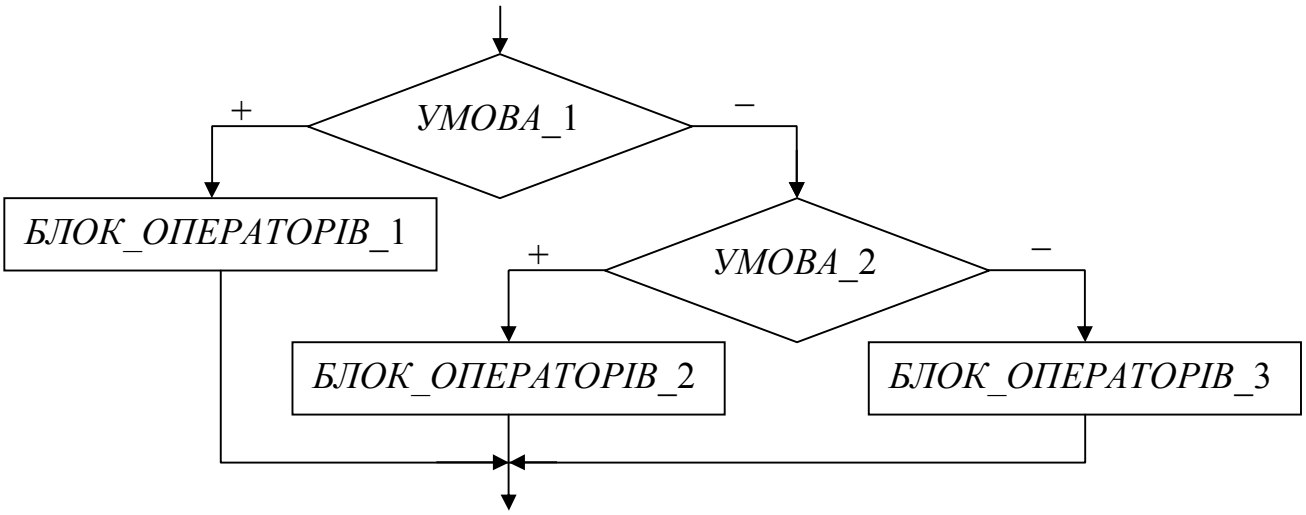
If *УМОВА*_1 Then
*БЛОК_ОПЕРАТОРІВ*_1
Else If *УМОВА*_2 Then
*БЛОК_ОПЕРАТОРІВ*_2
Else
*БЛОК_ОПЕРАТОРІВ*_3
End If
End If

У блок-схемі алгоритму такої послідовності операторів відповідає наступна конструкція

Для організації переходу в визначену точку програми використовується оператор безумовного переходу **GoTo**, що має наступний синтаксис:

**GoTo** *МІТКА*

*МІТКА* - це пойменована точка в тексті програми, у яку виконується безумовний перехід. Після мітки в програмі обов'язково ставиться двокрапка.




## 14.4. Приклад виконання лабораторної роботи «Організація лінійного і розгалуженого обчислювальних процесів»

**Завдання**. На основі блок-схеми алгоритму з лабораторної роботи (п. 6.4) розробити програму мовою VBA.

Рекомендації зі складання програми: вхідні дані вводяться за допомогою оператора InputBox, отримані результати виводяться за допомогою оператора MsgBox.

Програма рішення задачі на VBA.

```
Option Explicit
Public Sub prog1()
'Опис константи
Const Pi = 3.14159
'Опис змінних
Dim a As Single, b As Single
Dim x As Single, y As Single
'Ввід вхідних даних
a = InputBox("Введіть значення a", "Ввід вхідних даних")
b = InputBox("Введіть значення b", "Ввід вхідних даних")
'Обчислення значення X
If a * b > 1 Then
 x = Log(a * b) - 1
Else
 If a <> 0 Then
  x = (b - 1) / a
 Else
  'Вивод повідомлення про виникнення «аномалії» і перехід на мітку
  MsgBox "Ділення на 0", , "Помилка!"
  GoTo m1
```

```
 End If
End If
'Вивод значення X
MsgBox "x = " & x, , "Результати"
'Обчислення значення Y
If x < 1.5 Then
 y = Sqr(a ^ 2 + 1) + Sin(Pi / 2 * x)
Else
 If x >= 1.5 And x <= 3.5 Then
  y = Abs(a + x)
 Else
  If x - a >= 0 Then
   y = Sqr(x - a)
  Else
     'Вивод повідомлення про виникнення «аномалії» і перехід на мітку
    MsgBox "Корінь не існує", , "Помилка!"
    GoTo m1
  End If
 End If
End If
'Вивод значення Y
MsgBox "y = " & y, , "Результати"
m1:  'Мітка
End Sub
```

# Лекція №15. ОПЕРАТОРИ ЦИКЛУ. СТВОРЕННЯ ФОРМ. КЕРУЮЧІ ЕЛЕМЕНТИ

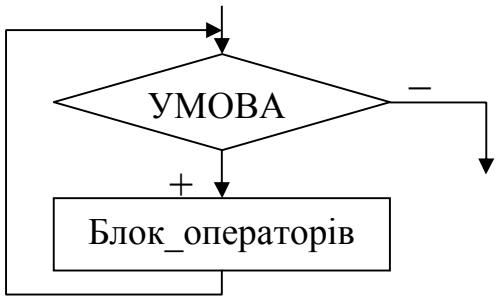
## 15.1. Оператор циклу Do … Loop

В VBA для організації циклів використовуються оператор циклу **-о …Loop**, який може бути реалізований у вигляді:

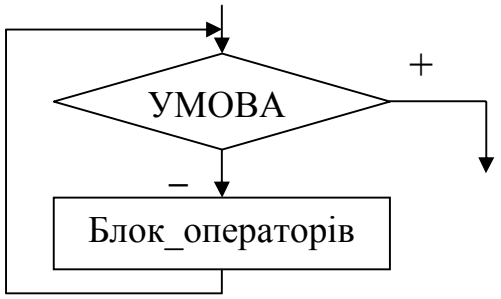
циклів з передумовою – **Do While … Loop,**
**Do Until … Loop;**
циклів з постумовою – **Do … Loop While,**
**Do … Loop Until**.

Нижче наведені синтаксис цих операторів циклу й показаний їхній принцип роботи у вигляді блок-схеми:

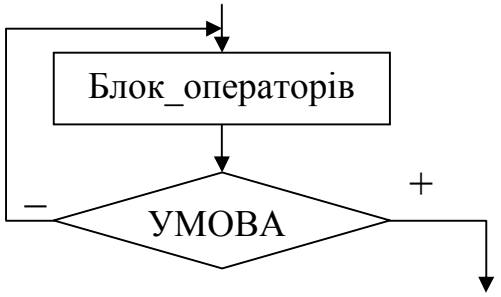
**Do While** *УМОВА*
  *БЛОК_ОПЕРАТОРІВ*
  [**Exit Do**]
  *БЛОК_ОПЕРАТОРІВ*
**Loop**



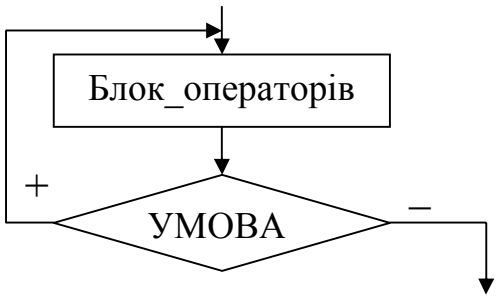

**Do Until** *УМОВА*
  *БЛОК_ОПЕРАТОРІВ*
  [**Exit Do**]
  *БЛОК_ОПЕРАТОРІВ*
**Loop**




**Do**
  *БЛОК_ОПЕРАТОРІВ*
    [**Exit Do**]
  *БЛОК_ОПЕРАТОРІВ*
**Loop Until** *УМОВА*




**Do**
  *БЛОК_ОПЕРАТОРІВ*
    [**Exit Do**]
  *БЛОК_ОПЕРАТОРІВ*
**Loop While** *УМОВА*




Оператор Do While…Loop забезпечує багаторазове повторення блоку операторів доти, поки УМОВА дотримується, а оператор Do Until … Loop поки УМОВА не дотримується. Оператори Do…Loop While, Do…Loop Until відрізняються від перерахованих вище операторів тим, що спочатку блок операторів виконується, принаймні, один раз, а потім перевіряється УМОВА. Для запобігання зациклення в тілі циклу повинен бути хоча б один оператор, який змінює значення змінних, які присутні в УМОВІ. Оператор **Exit Do** забезпечує достроковий вихід з оператора циклу**.**

**Приклад 15.1**. Перевірити, чи є задане ціле число простим.

Ціле число називається простим, якщо воно ділиться націло тільки на самого себе й одиницю. Алгоритм перевірки того, що число N є простим полягає в наступному (рис. 15.1): якщо N ділиться без залишку хоча б на одне число з діапазону від 2 до N/2, то число не є простим. Якщо немає жодного дільника числа, то число N – просте.

```
Public Sub prim1()
Dim N As Integer 'число, що перевіряється
Dim fl As Boolean 'fl=True – число є простим
Dim i As Integer
N = InputBox("Введіть значення N", "Ввід вхідних даних")
If N < 1 Then
 fl = False
Else
 fl = True
 i = 2
 'Початок циклу перевірки с постумовою
 Do
  If N Mod i = 0 Then  'перевірка остачі від ділення
   fl = False
   Exit Do  'достроковий вихід з циклу
  Else
   i = i + 1
  End If
 'умова виходу із циклу
 Loop Until i > N / 2
End If
If fl Then
  MsgBox "N = " & N _
  & " - просте число"
Else
  MsgBox "N = " & N _
  & " -  не просте число"
End If
End Sub
```

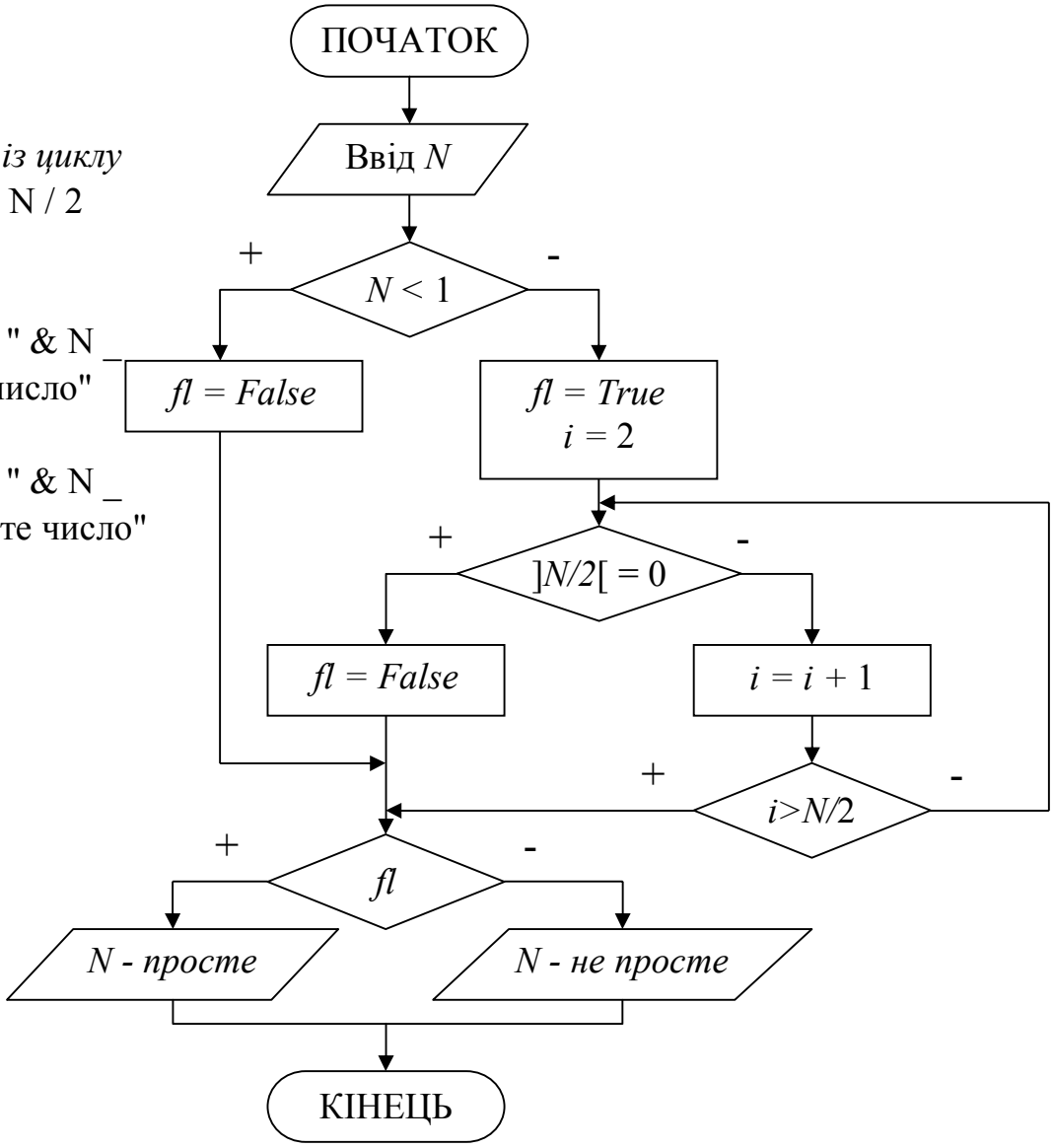



Рисунок 15.1. Алгоритм перевірки простого числа

## 15.2. Створення форм. Властивості, події й методи форм

***Форма*** – це головний об'єкт, що утворить візуальну основу додатка. По своїй суті форма являє собою вікно, у якому можна розміщати різні керуючі елементи при створенні додатків. Для створення форм необхідно виконати команду: **Вставка – UserForm**.

У вікні Конструктора форм з'явиться форма, що має стандартний вигляд для ОС Windows (рис. 15.2).


Рисунок 15.2. Вікно форми VBA

Як і будь-який інший об'єкт VBA форма має набір властивостей (табл. 15.1). Для одержання довідки по будь-якій властивості досить виділити її у вікні властивостей і натиснути F1.

Таблиця 15.1. Основні властивості форм

| **Властивість** | **Опис** |
|---|---|
| BackColor | Кольори фона для форми. |
| BorderStyle | Визначає тип границі, що оточує форму. |
| Caption | Текст, що виводиться в заголовку форми. |
| Font | Визначає тип і вигляд шрифту у формі. |
| Height | Визначає висоту форми у твіпах. |
| (Name) | Ім'я об'єкта, для програми VBA. |
| Width | Визначає ширину форми у твіпах. |

Властивості можна змінювати в режимі конструювання у вікні властивостей, або програмно в режимі виконання. Наприклад, у ході виконання програми можна змінити заголовок форми командою:

frmForm1.Captiton=”Лабораторна робота”

Програми в ОС Windows управляються ***подіями*** (табл. 15.2). Щораз, коли натискається кнопка, переміщається миша, змінюються розміри форми й тобто, ОС генерує повідомлення, що описує виконану дію, і поміщає його в чергу повідомлень програми. Із черги повідомлення доставляється відповідному об'єкту, наприклад формі, а та генерує відповідну подію. Отже, можна скласти фрагмент програми, у якому об'єкт буде реагувати на подію певним чином, тобто будь-якій стандартній події відповідає визначена процедура. Щоб переглянути події пов'язані з формою, необхідно в режимі конструювання двічі клацнути на ній – з'явиться вікно програми, у якому клацнути на списку Процедура.

Таблиця 15.2. Основні події форм

| Подія | Опис |
|---|---|
| Initialize | Відбувається під час конфігурації й до завантаження форми на згадку. |
| Activate | Відбувається після завантаження форми на згадку. |
| Deactivate | Відбувається, якщо форма перестає бути активною. |
| Click | Відбувається при натисканні лівої кнопки миші на формі. |

Наступний приклад змінює заголовок форми при активізації, і зменшує розмір форми після щиглика лівою кнопкою миші на формі.

```
Private Sub UserForm _Activate()
    frmForm1.Caption=”Щиглик на формі зменшує її розміри”
End Sub
Private Sub UserForm_Click()
    FrmForm1.Width= FrmForm1.Width/2
    FrmFotrm1.Height= FrmForm1.Height/2
    frmForm1.Caption=”Зроби це ще раз!”
End Sub
```

Також форма має набір методів і інструкцій. **Метод** визначає дію, що може бути виконане з об'єктом (табл. 15.3). **Інструкція** ініціює дію. Вона може виконати метод або функцію (табл. 15.4)**.**

Таблиця 15.3. Основні методи форм

| Метод | Опис |
|---|---|
| Hide | Приховує об'єкт UserForm, але не вивантажує його |
| Show | Виводить на екран об'єкт UserForm |

Таблиця 15.4. Основні інструкції форм

| Інструкція | Опис |
|---|---|
| Load | Завантажує об'єкт UserForm, але не відображає його на екрані. |
| Unload | Видаляє об'єкт UserForm з пам'яті. |

## 15.3. Вибір і використання елементів управління

Створення елементів управління на формі виконується за допомогою панелі інструментів, що виводиться на екран командою **Вид – Панель элементов (View – Toolbox)** (рис. 15.3).

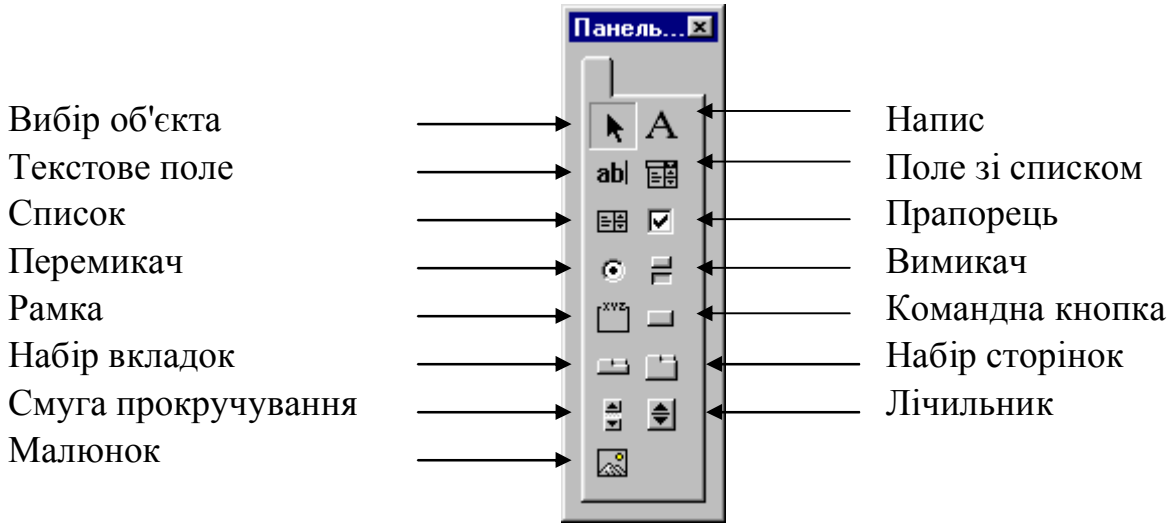



Рисунок 15.3. Панель елементів VBA.

За допомогою кнопок цієї панелі можна помістити у форму необхідний елемент управління. Для цього потрібно клацнути на значку елемента управління, далі при натиснутій лівій кнопці миші визначити розмір і місце розташування елемента у формі. Коли елемент на формі виділений (рамка об'єкта містить маленькі прямокутники) можна змінювати його розміри й переміщати за допомогою миші, а також переглядати й змінювати його властивості у вікні властивостей.

Кожний елемент управління (об'єкт) характеризується набором властивостей (які можна змінювати в режимах конструювання або виконання), подій і методів.

Для кожного об'єкта проекту необхідно визначити його ім'я. Відповідно до загальноприйнятих угод про імена об'єктів перші три символи імені повинні відображати вид елемента, а інші символи - призначення. У табл. 15.5. представлені сполучення перших трьох символів для елементів, які найбільш часто використовуються.

Таблиця 15.5. Сполучення перших трьох символів імен

| Об'єкт | Перші 3 символи імені | Приклад імені |
|---|---|---|
| Форма | frm | frmMyForm |
| Напис | lbl | lblInfo |
| Текстове поле | txt | txtInput |
| Командна кнопка | cmd | cmdExit |

***Командна кнопка*** є найпоширенішим елементом управління, і може використовуватися для організації виконання обчислень і інших дій, виклику процедур і функцій користувача, відкриття форм і т.д. Основні властивості командної кнопки представлені в табл. 15.6. У властивості **Caption** можна ставити символ & перед літерою, що буде використовуватися в сполученні із клавішею Alt для прискореного доступу до кнопки. Також можна перейти до кнопки кла-

вішею Tab, а потім натиснути Enter.

Таблиця 15.6. Властивості командних кнопок.

| Властивість | Опис |
|---|---|
| BackColor | Кольори фона кнопки. |
| Caption | Текст, що виводиться на кнопці. |
| Enabled | Значення False робить кнопку недоступної. |
| Font | Визначає тип і вид шрифту на кнопці. |
| ForeColor | Визначає кольори шрифту на кнопці. |
| (Name) | Ім'я об'єкта, для програми VBA. |
| Picture | Додає малюнок на кнопку. |
| PicturePosition | Визначає розташування тексту й малюнка на кнопці. |
| Visible | Значення False робить кнопку невидимої. |

Основною ***подією*** кнопки являється **Click**. Для написання програмного коду, що буде виконуватися при натисканні командної кнопки, досить два рази клацнути на ній лівою кнопкою миші в режимі конструювання проекту.

Найбільш корисним ***методом*** командної кнопки являється **SetFocus**, що дозволяє повернутися до кнопки (передати їй фокус). Наприклад, наступна команда дозволяє повернутися до кнопки за замовчуванням після вводу даних у текстове поле: cmdMyButtum.SetFocus.

***Текстове поле*** застосовується для вводу або виводу інформації. Основні властивості текстового поля представлені в табл. 15.7.

Таблиця 15.7. Властивості текстового поля.

| Властивість | Опис |
|---|---|
| Enabled | Значення False робить поле недоступним. |
| Font | Визначає тип і вигляд шрифту у текстовому полі. |
| ForeColor | Визначає кольори шрифту у текстовому полі. |
| (Name) | Ім'я об'єкта, для програми VBA. |
| MaxLength | Визначає кількість символів, що вводять, у текстове поле. |
| Text | Визначає вміст текстового поля. |

Наприклад, для очищення змісту текстового поля у процесі виконання програми потрібно ввести у необхідному місці програмного коду команду:

txtResult.Text=" "

Основною ***подією*** текстового поля є **Change**, що відбувається при введенні або видаленні символів. Наприклад, команду

cmdMyButtum**.**SetFocus

можна помістити у процедуру події Change текстового поля.

***Напис*** застосовується як самостійно для виводу довідкової інформації, так і у вигляді "підказок" для текстового поля, списку або іншого елемента. Головна її відмінність від текстового поля у тому, що користувач не може змінити текст напису (хоча його можна змінити як властивість під час виконання програми). Основні властивості напису представлені в табл. 15.8.

Таблиця 15.8. Властивості напису.

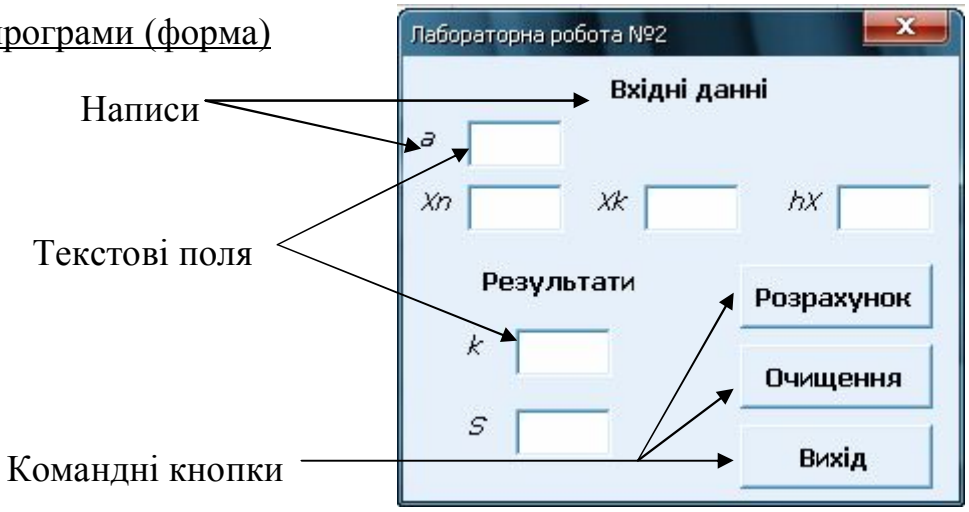
| **Властивість** | **Опис** |
|---|---|
| Caption | Визначає текст, що міститься у написі. |
| Font | Визначає тип і вигляд шрифту напису. |
| ForeColor | Визначає кольори шрифту напису. |
| (Name) | Ім'я об'єкта, для програми VBA. |

## 15.4. Приклад виконання лабораторної роботи «Організація циклів з відомим числом повторень»

**Завдання**. На основі блок-схеми алгоритму з лабораторної роботи (п. 7.3) розробити програму мовою VBA.

Рекомендації зі складання програми: для вводу вхідних даних створити форму з відповідними текстовими полями і кнопками, що керують роботою програми. Величини (*x, y*), значення яких виводяться в тілі циклу необхідно виводити на лист MS Excel у вигляді таблиці. Величини (*k, S*), значення яких виводяться поза тілом циклу, необхідно виводити у відповідні текстові поля на формі.

Робоче вікно програми (форма)




Програма рішення задачі на VBA.

*'Обчислення необхідних величин при натисканні кнопки "Розрахунок"*

```
Private Sub CmdSolve_Click()
```

```
Dim a As Single, xn As Single, xk As Single, hx As Single
Dim x As Single, y As Single, S As Single, k As Integer
```

*‘Ввід вхідних даних з текстових полів форми*

```
a = CSng(Txta.Text): xn = CSng(TxtXn.Text)
xk = CSng(TxtXk.Text): hx = CSng(Txth.Text)
x = xn: S = 0: k = 0: i = 2
Cells(1, 1) = "X"
Cells(1, 2) = "Y"
```

*'Початок циклу з пістумовою*

```
Do
 If x > 1 Then
  y = 0.5 * x + 0.6
 Else
  If x + a <> 0 Then
   y = a * Log(Abs(x + a))
  Else
   Cells(i, 1) = x
   Cells(i, 2) = "Логарифм не існує"
   GoTo m1
  End If
 End If
```

*'Вивід результатів (x,y) на лист Excel*

```
 Cells(i, 1) = x:   Cells(i, 2) = y
 If y > 1 Then S = S + y Else k = k + 1
m1:
 x = x + hx: i = i + 1
Loop Until x > xk
```

*'Вивід результатів (k,S) у текстові поля форми*

```
Txtk.Text = CStr(k): Txt.Text = CStr(S)
End Sub
```

*'Очищення всіх текстових полів форми і листа Excel*

```
'при натисканні кнопки "Очищення"
Private Sub CmdClear_Click()
Txta.Text = " ": TxtXn.Text = " "
TxtXk.Text = " ": Txth.Text = " "
Txt.Text = " ":  Txtk.Text = " "
Sheets(1).Range("A1:B100").Clear
End Sub
'Завершення роботи програми при натисканні кнопки "Вихід"
Private Sub CmdExit_Click()
End
End Sub
```

Лист Excel з результатами:

| | A | B |
|---|---|---|
| 1 | X | Y |
| 2 | -3 | 0 |
| 3 | -2 | Логарифм не існує |
| 4 | -1 | 0 |
| 5 | 0 | 1,386 |
| 6 | 1 | 2,197 |
| 7 | 2 | 1,600 |
| 8 | 3 | 2,100 |

## 15.5. Приклад виконання лабораторної роботи «Організація вкладених циклів»

**Завдання**. На основі блок-схеми алгоритму з лабораторної роботи (п. 8.4) розробити програму мовою VBA.

Рекомендації зі складання програми: робоче вікно програми буде містити тільки кнопки, що керують роботою програми. Ввід початкових даних виконується за допомогою операторів InputBox, після натискання кнопки «Ввід даних». Величини (*x, y*) будуть обчислюватися при натисканні кнопки «Розрахунок» і виводитися на лист MS Excel у вигляді таблиці. Змінні *an, ak, ha, bn, bk, hb* повинні бути ***глобальними***, тому що вони використовуються у двох процедурах (при вводі даних і при розрахунку) і не повинні губити свої значення при завершенні роботи процедури. Інші змінні будуть ***локальними***.

Робоче вікно програми (форма):

Програма рішення задачі на VBA.

```
'Опис глобальних змінних
Dim an As Single, ak As Single, ha As Single
Dim bn As Single, bk As Single, hb As Single

'Ввід вхідних даних при натисканні кнопки "Ввід даних"
Private Sub CmdInput_Click()
an=InputBox("Введіть значення an", "Ввід вхідних даних")
ak=InputBox("Введіть значення ak", "Ввід вхідних даних")
ha=InputBox("Введіть значення ha", "Ввід вхідних даних")
bn=InputBox("Введіть значення bn", "Ввід вхідних даних")
bk=InputBox("Введіть значення bk", "Ввід вхідних даних")
hb=InputBox("Введіть значення hb", "Ввід вхідних даних")
End Sub

'Обчислення необхідних величин  при натисканні кнопки "Розрахунок"
Private Sub CmdSolve_Click()
'Опис локальних змінних
Dim a As Single, b As Single
Dim x As Single, y As Single
Dim i As Integer

b = bn: i = 2
Cells(1, 1) = "B": Cells(1, 2) = "X"
Cells(1, 3) = "A": Cells(1, 4) = "Y"
'Початок зовнішнього циклу із передумовою
Do While b <= bk
 x = Cos(b) + 0.2
 Cells(i, 1) = b
 Cells(i, 2) = x
 a = an
 'Початок внутрішнього циклу із передумовою
 Do While a <= ak
  If x < 0.6 Then
   y = x ^ 2 + Abs(a - b)
  Else
   If a - x <> 0 Then
    y = a * b - x / (a - x)
   Else
    Cells(i, 3) = a:
```

```
    Cells(i, 4) = "Ділення на 0"
    GoTo m1
   End If
  End If
  Cells(i, 3) = a
  Cells(i, 4) = y
m1:
  a = a + ha
  i = i + 1
 Loop 'Завершення внутрішнього циклу
 b = b + hb
Loop 'Завершення зовнішнього циклу
End Sub
```

*'Очищення листа Excel при натисканні кнопки "Очищення"*

```
Private Sub CmdClear_Click()
Sheets(1).Range("A1:D100").Clear
End Sub
```

Лист Excel з результатами:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **B** | **X** | **A** | **Y** |
| 2 | 2 | -0,21615 | 1 | 1,046719 |
| 3 | | | 2 | 0,046719 |
| 4 | | | 3 | 1,046719 |
| 5 | | | 4 | 2,04672 |
| 6 | 4 | -0,45364 | 1 | 3,205792 |
| 7 | | | 2 | 2,205792 |
| 8 | | | 3 | 1,205793 |
| 9 | | | 4 | 0,205793 |
| 10 | 6 | 1,16017 | 1 | 13,24335 |
| 11 | | | 2 | 10,61856 |
| 12 | | | 3 | 17,36941 |
| 13 | | | 4 | 23,59146 |

# Лекція №16. МАСИВИ У VBA. РОБОТА З ФАЙЛАМИ

## 16.1. Опис масивів

Масив описується наступною конструкцією:

**Dim** *ІМ'Я_МАСИВУ*(Розмірність) **As** *ТИП_МАСИВУ*

Наприклад:

Dim A(12) As Integer – оголошується вектор з 12 цілих чисел, причому за замовчуванням перший елемент масиву буде A(0), а останній A(11). Говорять, що 0 – **базовий індекс**.

Dim B(3,3) As Single – оголошується матриця 3×3 дійсних чисел.

Щоб задати масив, у якого 1-й елемент буде мати індекс 1, необхідно використовувати наступну конструкцію:

**Dim** *ІМ'Я_МАСИВУ*(1 **to** Розмірність) **As** *ТИП_МАСИВУ*

Наприклад:

Dim B(1 to 3, 1 to 3) As Single – використання ключового слова **to** при оголошенні масиву змінює базовий індекс на 1.

Масив у програмі ініціалізується по елементам, наприклад:

Dim B (1 to 2) As Single

B(1) = 2.1: B(2) = 4.6

Іноді розмір масиву не може бути визначений заздалегідь. У такому випадку можна оголосити масив, який називається **динамічним**, без зазначення розмірності

**Dim** *ІМ'Я_МАСИВУ*() **As** *ТИП_МАСИВУ*

Наприклад: Dim Y() As Integer

Потім додавання елементів у масив здійснюється оператором **ReDim**, який можна використовувати тільки в процедурах.

Наприклад: ReDim Y(5).

Елементам, які створені, необхідно присвоїти значення. Пізніше кількість елементів масиву знову можна змінити. При цьому всі значення, що зберігаються в масиві, втрачаються. Щоб зберегти їх необхідно використовувати ключове слово **Preserve**.

Наприклад: Preserve ReDim Y(7).

## 16.2. Оператор циклу For

Для організації циклу «Для» в VBA служить оператор For...Next, що має наступний синтаксис:

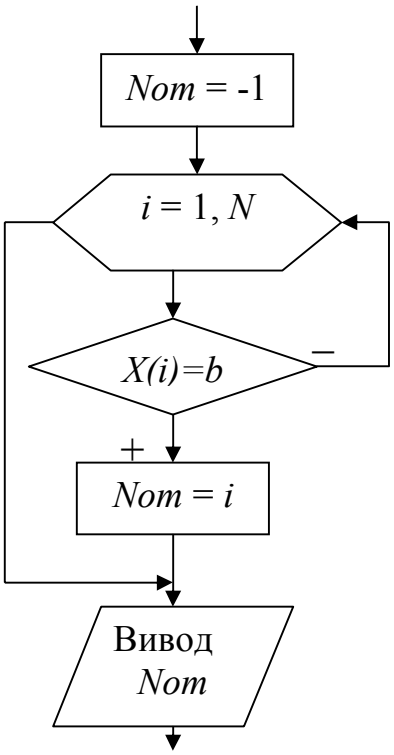
**For** *ЛІЧИЛЬНИК=ПОЧ_ЗНАЧЕННЯ* **To** *КІН_ЗНАЧЕННЯ* **Step** *КРОК*
*БЛОК_ОПЕРАТОРІВ*
[**Exit For**]
*БЛОК_ОПЕРАТОРІВ*
**Next** *ЛІЧИЛЬНИК*

Цикл For…Next перебирає значення змінної *ЛІЧИЛЬНИК*, яка є параметром циклу, від початкового до кінцевого значення із указаним кроком зміни. При цьому забезпечується виконання блоку операторів тіла циклу при кожному новому значенні лічильника. Якщо **Step** *КРОК* у конструкції відсутній, то за замовчуванням вважається, що крок дорівнює 1. По оператору **Exit For** можна вийти з оператора циклу до того, як *ЛІЧИЛЬНИК* досягне останнього значення.

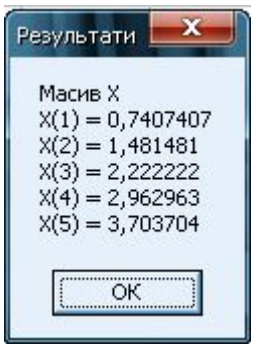
**Примітка.** Не рекомендується примусово змінювати значення параметра циклу, його початкового й кінцевого значення в тілі циклу For...Next.

**Приклад 16.1**. Визначити номер першого входження елемента *b* у вектор *X*. Якщо серед компонентів вектора *X* немає елементів, рівних *b*, то результату

присвоїти значення рівне –1. Фрагменти блок-схеми й програми, що реалізують поставлену задачу, можуть мати такий вигляд:




```
'Формування елементів масиву X і
'одержання значення b
...
Nom = -1
For i = 1 To N
  If X(i) = b then Nom = i: Exit For
Next i
If Nom = -1 Then
  MsgBox "Вектор X не містить b", _
   vbOKOnly, "Результат"
Else
  MsgBox "Номер першого входження  = "_
   & CStr(Nom), vbOKOnly, "Результат"
End If
...
```




**Приклад 16.2**. Виконати вивід елементів одномірного масиву за допомогою однієї функції MsgBox.

```
'Опис строкової змінної St для зберігання
'виведеної інформації
Dim St As String
...
'Формування елементів масиву
...
'Присвоювання початкового значення змінної St
St = "Масив X" & Chr(13)
'Заповнення змінної St значеннями елементів масиву X
For i = 1 To N
  St =  St & "X(" & CStr(i) & ")= " & CStr(x(i)) &
hr(13)
Next i
'Вивід змінної St, яка містить масив X
MsgBox St, vbOKOnly, "Результати"
```

**Приклад 16.3**. Скласти процедуру **Переведення**, що виконує переведення цілого десяткового числа у двійкову систему числення.

```
Public Sub Переведення()
Dim a As Integer, b As Byte, i As Integer
Dim st As String
'Масив для зберігання двійкових цифр числа
Dim x(1 To 8) As Byte
a = InputBox("Введіть ціле число >0 і <256","Ввід даних")
'Перевірка правильності введеного числа
If a >= 0 And a <= 255 Then
 i = 0: b = a
 'Цикл із постумовою для переведення числа у двійкову
 'систему числення
 Do
  i = i + 1
  x(i) = b Mod 2
  b = b \ 2
 Loop Until b = 0
 b = i
 st = "Двійкове представлення числа " & CStr(a) & Chr(13)
 'Вивід елементів масиву у зворотному порядку
 For i = b To 1 Step -1
   st = st & CStr(x(i)) & " "
 Next i
 MsgBox st, vbOKOnly, "Результат"
Else
 MsgBox "Введене число не із зазначеного діапазону!",_
         vbOKOnly, "Помилка"
End If
End Sub
```

Для перебору об'єктів із групи подібних об'єктів, наприклад, комірок з діапазону або елементів масиву, зручно використовувати оператор циклу **or…Each…Next...**

**For Each** *ЕЛЕМЕНТ* **In** *ГРУПА*
    *БЛОК_ОПЕРАТОРІВ*
    **[Exit For]**
    *БЛОК_ОПЕРАТОРІВ*
**Next** *ЕЛЕМЕНТ*

**Приклад 16.4.** Складемо процедуру **Знак,** що заміняє всі додатні числа з діапазону комірок A1:B2 знаком “+”, всі від’ємні числа – знаком “–“, а нулі залишає без зміни.

```
Option Explicit
Public Sub Знак( )
Dim c As Object
For Each c In Worksheets("Лист1").Range("A1:B2")
  If IsNumeric(c.Value) Then
     If c.Value > 0 Then
        c.Value = "+"
     ElseIf c.Value < 0 Then
            c.Value = "-"
         Else
            c.Value = 0
     End If
  End If
Next c
End Sub
```

У процедурі **Знак** використовувалася функція **IsNumeric(***ВИРАЗ***), яка** повертає True, якщо *ВИРАЗ* може бути описане як числовий, і False у противному випадку.

### 16.3. Запис і читання даних у файли

При роботі з великою кількістю даних часто буває зручно записувати дані у файл або зчитувати з файлу. Робота з файлом будь-якого типу починається з його відкриття. Для цього використовується оператор **Open**, що відкриває файл і готує його до читання або запису.

Синтаксис оператору:

**Open** *ІМ'Я_ФАЙЛУ* **For** *РЕЖИМ* **[Access** *ДОСТУП*] *[БЛОКУВАННЯ*]
**As** [#]*НОМЕР_ФАЙЛУ* **[Len=***ДОВЖИНА***]**

Опис параметрів оператора Open наведене в табл. 16.1.

Таблиця 16.1. Параметри оператора Open.

| **Параметр** | **Опис** |
|---|---|
| **1** | **2** |
| *ІМ'Я_ФАЙЛУ* | Обов'язковий. Строковий вираз, що вказує ім'я файлу; може містити ім'я каталогу або папки й ім'я диска. |
| *РЕЖИМ* | Обов'язковий. Ключове слово, що вказує режим відкриття файлу: **Append**, **Binary**, **Input**, **Output** або **Random**. За замовчуванням, файл відкривається для доступу в режимі Random. |
| *ДОСТУП* | Необов'язковий. Ключове слово, що вказує операції, які дозволені з відкритим файлом: **Read**, **Write** або **Read Write**. |

Продовження табл. 16.1

| 1 | 2 |
|---|---|
| *БЛОКУВАННЯ* | Необов'язковий. Ключове слово, що вказує операції, дозволені з відкритим файлом іншим процесам: **Shared**, **Lock Read**, **Lock Write** і **Lock Read Write**. |
| *НОМЕР_ФАЙЛУ* | Обов'язковий. Припустимий номер файлу в інтервалі від 1 до 511 включно. Для визначення наступного вільного номера файлу варто використовувати функцію **FreeFile**. |
| *ДОВЖИНА* | Необов'язковий. Число, менший або рівне 32767 (байт). Для файлів, відкритих у режимі Random, це значення є довжиною запису. Для файлів з послідовним доступом це значення є числом буферізуємих символів. |

В VBA забезпечується три типи доступу до файлів:

– послідовний доступ (режими **Input**, **Output** и **Append**), звичайно використовується для запису текстових файлів, наприклад протоколів помилок або звітів;

– довільний доступ (режим **Random**), який використовується при необхідності зчитати і записати дані у файл без його закриття. Файли довільного доступу містять дані в виді записів, котрі спрощують і прискорюють пошук потрібних відомостей;

– двійковий доступ (режим **Binary**), який використовується коли потрібно зчитати або записати байт у кожну позицію в файлі, наприклад при збереженні або відображенні крапкових зображень.

Інструкцію **Open** не слід використовувати для доступу до власних типів файлів додатків. Наприклад, не слід використовувати Open для доступу до документа Word, до електронної таблиці Microsoft Excel або до бази даних Microsoft Access, оскільки це спричинить втрату цілісності й псую файлу.

Якщо аргумент *ІМ'Я_ФАЙЛУ* описує неіснуючий файл, такий файл буде створений при відкритті в режимі Append, Binary, Output або Random. Якщо файл уже відкритий іншим процесом і указаний режим доступу не дозволений, інструкція Open не буде виконана й виникне помилка. У режимах Binary, Input і Random можна ще раз відкрити вже відкритий файл під іншим номером, не закриваючи його. У режимі Append і Output необхідно закрити файл, щоб отримати можливість відкрити його ще раз під іншим номером.

Для запису даних у послідовний файл необхідно відкрити його в режимі Output (при цьому дані, наявні у файлі, стираються) або Append (нові дані додаються в кінець файлу). Для читання даних з послідовного файлу варто відкрити його в режимі Input.

Після завершення роботи з файлом його варто закрити оператором **Close**.

**Close** [#]*НОМЕР_ФАЙЛУ*

Нижче приведений фрагмент програми, що створює в папці *C:\Data* файл послідовного доступу *test.txt* для виводу даних:

```
Dim FileNo As Integer
'Отримати унікальний файловий номер
FileNo = FreeFile
Open "C:\Data\test.txt" For Output As FileNo
...
БЛОК_ОПЕРАТОРІВ
...
Close FileNo
```

У таблиці 16.2 наведені оператори, звичайно використовувані для запису даних у файли й для читання даних з файлів.

Таблиця 16.2.Оператори зчитування й запису даних у файли.

| **Тип доступу** | **Запис даних** | **Читання даних** |
|---|---|---|
| Послідовний | Print #, Write # | Input # |
| Довільний | Put | Get |
| Двійковий | Put | Get |

Оператор **Print** # записує відформатовані дані у файл із послідовним доступом. Синтаксис:

**Print** # *НОМЕР_ФАЙЛУ[, СПИСОК_ВИВОДУ]*

*НОМЕР_ФАЙЛУ* є обов'язковим параметром. *СПИСОК_ВИВОДУ* необов'язковий параметр і складається з виразу або списку виразів, які варто вивести у файл. Нижче наведені припустимі значення аргументу *СПИСОК_ВИВОДУ* і їхній опис (табл. 16.3.):

*[{**Spc**(n) | **Tab**[(n)]}] [ВИРАЗ] [ПОЗИЦІЯ]*

Таблиця 16.3. Опис елементів списку виводу.

| **Значення** | **Опис** |
|---|---|
| Spc(n) | Використовується для вставки пробілів у файл; тут n число пробілів, які варто вставити. |
| Tab(n) | Встановлює курсор у колонку із зазначеним номером; тут n номер колонки. Tab без аргументу встановлює курсор у початок наступної зони виводу. |
| *ВИРАЗ* | Числові вирази або строкові вирази, які варто вивести. |
| *ПОЗИЦІЯ* | Указує позицію, у якій варто вивести наступний символ. Для установки курсору відразу після останнього виведеного символу використовуйте крапку з комою. Для установки курсору в колонку із зазначеним номером використовуйте Tab(n). Якщо аргумент позиція опущена, наступний символ виводиться на наступному рядку. |

Якщо параметр *СПИСОК_ВИВОДУ* опущений, після параметра *НОМЕР_ФАЙЛУ* йде тільки роздільник списку, у файл друкується порожній рядок. Для розподілу виразів можна використовувати пробіли або крапки з комою, які в даній ситуації цілком еквівалентні.

Дані, записані за допомогою оператора **Print #**, звичайно зчитуються з файлу за допомогою операторів **Line Input #** або **Input.** Для запису у файл даних, що у майбутньому планується читати за допомогою оператора **Input #**, треба замість оператора **Print #** використовувати оператор **Write #**, що має аналогічний синтаксис. Використання оператора **Write #** гарантує, що записані дані будуть коректно розділені, що дозволить прочитати їх за допомогою оператора **Input #**.

Оператор **Input #** читає дані з відкритого послідовного файлу й присвоює їх змінним. Синтаксис:

**Input #** *НОМЕР_ФАЙЛУ, СПИСОК_ЗМІННИХ*

*СПИСОК_ЗМІННИХ* – це поділюваний комами список змінних, котрим варто присвоїти значення, зчитані з файлу. Не можна використовувати масиви або об'єктні змінні. Однак допускається використання змінних, що описують елементи масиву або обумовленого користувачем типу.

Дані, які зчитали за допомогою інструкції **Input #**, звичайно записуються у файл за допомогою інструкції **Write #**. Ця інструкція застосовна тільки до файлів, які відкрити у режимі **Input** або **Binary**. Після зчитування стандартні рядки й числові дані присвоюються змінним без зміни.

При досягненні кінця файлу під час зчитування елемента даних ввід припиняється й виникає помилка. Щоб мати можливість коректно зчитувати дані з файлу в змінні за допомогою інструкції Input #, варто завжди використовувати інструкцію Write # (а не Print #) для запису даних у файли. Використання інструкції Write # гарантує правильність розміщення роздільників між окремими елементами даних.

Для перевірки досягнення кінця файлу при зчитуванні елемента даних використовується функція **EOF**.

**EOF (**[#]*НОМЕР_ФАЙЛУ*)

Нижче наведений фрагмент програми, у якому в масив рядків завантажується вміст файлу C:\Data\test.txt до повного його вичерпання:

```
Dim FileNo As Integer
'Опис динамічного масиву GetValues строкового типу
Dim GetValues() As String
Dim i As Integer
'Ініціалізація лічильника і
i = 0
FileNo = FreeFile
Open "C:\Data\test.txt" For Input As FileNo
```

```
 'Виконання циклу поки не досягнуть кінець файлу
  Do Until EOF(FileNo)
     i = i +1
     'Перевизначення масиву GetValues розмірності і
     'зі збереженням раніше привласнених значень
     ReDim Preserve GetValues(i)
     Input #FileNo, GetValues(i)
   Loop
Close FileNo
```

## 16.4. Приклад виконання лабораторної роботи «Обробка одномірних масивів з перестановкою елементів»

**Завдання**. На основі блок-схеми алгоритму з лабораторної роботи (п. 11.6) розробити програму мовою VBA.

Рекомендації зі складання програми: Початкові дані (розмірність і значення елементів масиву *X*) вводяться з листа Excel. Результати (значення елементів масиву *Y*) виводяться на лист Excel.

```
Public Sub prog7()
'Опис масивів
Dim x(10) As Single, y(10) As Single
Dim i As Integer, N As Integer
'Ввід розмірності
N = Cells(1, 2)
k = Cells(1, 5)
'Ввід елементів початкового масиву
For i = 1 To N
 x(i) = Cells(3, i)
Next i
Cells(4, 1) = "Масив Y"
'Формування масиву Y і вивід його елементів
For i = 1 To N
  If x(i) <= k Then
   y(i) = x(i + N - k)
  Else
   y(i) = x(i - k)
  End If
  Cells(5, i) = y(i)
Next i
End Sub
```

Лист Excel з початковими даними та результатами:

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N = | 10 | | k = | 3 | | | | | |
| 2 | Масив X | | | | | | | | | |
| 3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 4 | Масив Y | | | | | | | | | |
| 5 | 8 | 9 | 10 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

## 16.5. Приклад виконання лабораторної роботи «Обробка двовимірних масивів»

**Завдання**. На основі блок-схеми алгоритму з лабораторної роботи (п. 12.3) розробити програму мовою VBA.

Рекомендації зі складання програми: Початкові дані (розмірність і значення елементів масиву *A*) вводяться з листа Excel. Результати (значення елементів масиву *B*) виводяться на лист Excel.

```
Public Sub prog8()
'Опис масивів
Dim A(10, 10) As Single, B(10) As Single
Dim M As Integer, N As Integer
Dim i As Integer, j As Integer
'Ввід розмірності  матриці
M = Cells(1, 2)
N = Cells(1, 5)
'Ввід елементів вхідної матриці
For i = 1 To M
 For j = 1 To N
  A(i, j) = Cells(i + 2, j)
 Next j
Next i
Cells(M + 4, 1) = "Масив B"
'Формування масиву B и вивід його елементів
For j = 1 To N
 B(j) = 0
 For i = 1 To M
  B(j) = B(j) + A(i, j)
 Next i
 B(j) = B(j) / M
 Cells(M + 5, j) = B(j)
Next j
End Sub
```

Лист Excel з початковими даними та результатами

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | M = | 4 | | N = | 5 |
| 2 | Масив A | | | | |
| 3 | 5 | 0 | 6 | 0 | 7 |
| 4 | 2 | 8 | -4 | 8 | -2 |
| 5 | 6 | -4 | 3 | 5 | 11 |
| 6 | 1 | 2 | -1 | -3 | 6 |
| 7 | | | | | |
| 8 | Масив B | | | | |
| 9 | 3,5 | 1,5 | 1 | 2,5 | 5,5 |

# Лекція №17. ОСНОВИ РОБОТИ В МЕРЕЖІ INTERNET

## 17.1. Структура Internet

***Internet*** – це мережа зв'язаних один з одним комп'ютерних мереж. Оскільки всяка мережа являє собою певну кількість зв'язаних комп'ютерів, то Internet можна представити також як величезну кількість зв'язаних один з одним комп'ютерів.

З організаційної точки зору, мережа Інтернет не належить нікому, а представляє об'єднання різних регіональних і корпоративних ділянок мережі, кожний з яких управляється своєю організацією по своїх внутрішніх правилах. Існує, однак, ряд організацій, в основному суспільних, які виконують конкретні функції, що дозволяють підтримувати порядок у мережі, виробляють загальні для всіх рекомендації, сприяють поширенню нових технологій.

Мережа Інтернет поєднує комп'ютери й локальні мережі, які розташовані в різних населених пунктах і країнах, для чого використовують не тільки окремо виділені лінії зв'язку (оптоволоконні або супутникові), але й телефонні, і телеграфні канали загального призначення.

Для підключення комп'ютерів користувачів локальної мережі до мережі Інтернет створюється шлюз (спеціально виділений комп'ютер або комутаційний пристрій), який підключений через зовнішній канал зв'язку до комп'ютера сервіс-провайдера – організації, що займається наданням послуг з доступу до глобальної мережі. Ці комп'ютери сервіс-провайдера, а часто й комп'ютери-шлюзи користувачів постійно підключені до мережі Інтернет. Вони отримали назву серверів (хостів, вузлів) мережі.

Користувачі, як правило, підключаються безпосередньо до серверів провайдера по постійним або лініям зв'язку, що комутируються.

Надійність обміну інформацією при формуванні ліній зв'язку мережі забезпечують кілька каналів доступу до серверів мережі. У цьому випадку зв'язок між комп'ютерами може здійснюватися по декількох напрямках, з використанням різних проміжних вузлів мережі.

Для керування рухом інформації з різних ділянок мережі застосовується різноманітне комутаційне обладнання: маршрутизатори, мости й т.д. Ці пристрої визначають маршрут проходження інформації у мережі Інтернет і виключають влучення внутрішньої інформації локальних мереж в Інтернет і навпаки.

## 17.2. Основні сервіси мережі Internet

Додатками або видами сервісу йменують окремі можливості або способи використання глобальних мереж. Основними сервісами **Internet** не сьогоднішній день є наступні:

**Всесвітня павутина** (WWW – World Wide Web) – у цей час цей базовий додаток Інтернет, що забезпечує доступ до гігантського обсягу інформації у всіх можливих формах: текст, графіка, анімація, звук, відео. В основі WWW лежить технологія гіпертексту (текст плюс зв'язки (гіперпосилання на) з іншими текстами, графікою, відео або звуковою інформацією). Винайдена на початку 1990 років мова розмітки гіпертексту – HTML (HyperText Markup Language) – дозволяє легко включати в HTML – документи, з яких в основному й складається WWW, самі різні типи об'єктів і робити посилання на інші файли, поза залежністю від того, де вони знаходяться. Одиницею гіпертексту є web-сторінка – мінімальний документ, котрий можна завантажити й прочитати за один раз. Сукупність web-сторінок становить web-сайт. Передача даних у Всесвітній павутині здійснюється завдяки протоколу передачі гіпертексту – HTTP (HyperText Transmission Protocol). Програми-клієнти WWW називаються браузерами. Web-браузер перекладає інформацію до виду, доступній людині.

**Електронна пошта** (E-mail) – служить для пересилання електронних повідомлень конкретним одержувачам за допомогою поштових серверів. Доступ до послуги здійснюється по протоколах електронної пошти за допомогою клієнтських програм, наприклад, Outlook Express, Eudora, The Bat.

**Файлові архіви** (FTP – File Transfer Protocol) – служить для пересилання файлів з одного комп'ютера на іншій. У систему іншого комп'ютера можна ввійти, знаючи потрібне ім'я й пароль, при цьому можливий і анонімний доступ.

**Телеконференції** (групи новин) дозволяють вести через мережу дискусії по темах, що цікавлять Вас. На відміну від електронної пошти повідомлення телеконференції групуються по певних темах і посилають не індивідуальним користувачам, а поміщаються в групи новин на сервері новин. Доступ до послуги здійснюється за допомогою клієнтської програми електронної пошти, наприклад, Outlook Express, по протоколах обміну повідомленнями в телеконференціях NNTP.

**Списки розсилання** використовуються для розсилання повідомлень по визначених темах за допомогою електронної пошти. Доступ до послуги здійснюється за допомогою клієнтської програми електронної пошти, наприклад, Outlook Express, по протоколах електронної пошти.

**Online засоби комунікації користувачів** забезпечують в Інтернет бесіду, що чергується, обмін текстовими повідомленнями в реальному часі. Доступ до послуги здійснюється за допомогою протоколу IRC (Internet Relay Chat).

## 17.3. Протоколи Internet

Комп'ютери, що підключені до Internet, мають різну архітектуру, на них установлене різне програмне забезпечення. При роботі в мережі сумісність да-

них досягається за рахунок використання комунікаційних протоколів - правил передачі інформації з мереж. Робота Internet заснована на використанні протоколу TCP/IP (Transmission Control Protocol/Internet Protocol), під яким мається на увазі безліч протоколів, кожний з яких вирішує свій набір завдань:

- транспортний протокол TCP управляє процесом передачі даних між комп'ютерами;
- протоколи маршрутизації IP, ICMP (Internet Control Message Protocol), RIP (Routing Information Protocol) обробляють адресацію даних, забезпечують їхню фізичну передачу й відповідають за вибір найкращого маршруту до адресата;
- протоколи підтримки мережної адреси DNS (Domain Name System), ARP (Address Resolution Protocol) забезпечують ідентифікацію комп'ютера по її унікальній адресі;
- шлюзові протоколи EGP (Exterior Gateway Protocol), GCP (Gateway-to-gateway protocol), IGP (Interior Gateway Protocol) відповідають за передачу інформації про маршрутизації даних і стану мережі, а також обробляють дані для взаємодії з локальними мережами;
- протоколи прикладних сервісів FTP (File Transmission Protocol), Telnet і ін. - мережні програми, що забезпечують доступ до різних послуг і служб Мережі- наприклад, передачі файлів між комп'ютерами;
- інші важливі протоколи, наприклад, протокол SMTP (Simple Mail Transfer Protocol) відповідає за передачу повідомлень електронної пошти.

## 17.4. Адресація в Internet

Усякий комп'ютер, який підключений до мережі Internet, має унікальну адресу, яка називається *IP-адресою* («ай-пи»-адреса). IP-адреса машини може бути постійною або призначатися сервером при з'єднанні. Усякий хост завжди має одну IP-адресу.

При роботі в Internet доменні імена заміняються на зрозумілі комп'ютеру IP-адреси за допомогою системи DNS (Domain Name System), що представляє собою ієрархічну систему DNS-серверів.

*Доменне ім'я* є аналогом IP-адреси з тією лише різницею, що для зручності користувачів замість цифр, розділених крапками, у ньому використовуються слова, які розділені крапками. Доменне ім'я складається з декількох ієрархічно розташованих *доменів*, під якими мають на увазі ієрархічний набір вузлів, що об'єднані по територіальній або організаційній ознаці.

Правила складання доменних імен менш тверді в порівнянні з IP-адресами. Доменне ім'я записується праворуч ліворуч за правилом ієрархічного підпорядкування доменів.

Наприклад, доменне ім'я сервера ДонНТУ **donntu.edu.ua** містить у собі наступні частини:

**donntu**- *домен третього рівня* – ім'я організації, у цьому випадку латиномовний аналог ДонНТУ;

**edu-домен** *другого рівня* – у цьому випадку вказує на приналежність до освітньої мережі;

**ua-домен** *верхнього рівня* – у цьому випадку головний домен України.

Домени верхнього рівня бувають двох типів: географічні (двобуквені - кожній країні відповідає двобуквений код) і адміністративні (трибуквені).

Наприклад, домен України – **ua**, у Росії їх два – **ru** (від Russia) і домен який залишився від СРСР – **su** (від USSR). Список деяких головних доменів наведений у табл. 17.1.

Таблиця 17.1. Список основних доменів

| Країна | Домен | Тип організації | Домен |
|---|---|---|---|
| Україна | Ua | Комерційна | com |
| Росія | ru, su | Освітня | edu |
| Великобританія | Uk | Урядова | gov |
| Франція | Fr | Комп'ютерна мережа | net |
| Німеччина | De | Некомерційна | org |
| США | us | Міжнародна | int |

Система доменних імен, однак, виявляє лише основу системи адресації. Кожний розміщений в Інтернет документ має власну адресу, який позначуваний як URL (Uniform Resource Locator) – єдиний вказівник ресурсу. URL, крім вказівки доменного ім'я, включає також і вказівку шляху до конкретної сторінки.

Електронна адреса має чітко задану структуру. Всі символи набираються без пробілу.

метод://ім'я_сервера.ім'я_домена:порт/ім'я_каталогу/ім'я_файлу

Під методом розуміється спосіб передачі даних, тобто який-небудь протокол. Це може бути протокол http, gopher, telnet, ftp і ін.

Сайти, здебільшого, мають досить розгалужену ієрархічну структуру, основу якої становлять численні директорії, які поділювані косими рисами, крапкою, тире й іншими символами. Тому адреса конкретного документа, як правило, має вигляд, подібний до наведеного документа із сервера ДонНТУ:

http://donntu.edu.ua/russian/strukt/institutes/igg/home.html

У цьому випадку

http://donntu.edu.ua – вказівка сайту,

/russian/strukt/institutes/igg - вказівка шляху до файлу (фактично – перелік

директорій),

/home.html - ім'я конкретного файлу.

Варто пам'ятати, що при вводі адреси вручну помилка навіть в одному символі критична. Із цієї причини не рекомендується переносити (записувати) складні адреси на папері й потім уводити їх вручну із клавіатури – імовірність помилки в цьому випадку дуже велика.

## 17.5. Механізм обробки запиту

Обслуговуванням процесу передачі даних займаються програми, які можуть виконуватися як на комп'ютері клієнта, так і на віддаленому комп'ютері-сервері. Перші за аналогією називаються програмами-клієнтами або просто клієнтами, другі – програмами-серверами або просто серверами. Клієнтами є, наприклад, браузери, за допомогою яких користувач переглядає запитані Web-сторінки на своєму комп'ютері. Сервером є, наприклад, proxy-сервер – програма, що приймає посередницьку участь між клієнтським комп'ютером і фізичним сервером.

Запит на з'єднання може бути направлений адресатові у вигляді IP-адреси або доменного імені від комп'ютера, що має свою адресу. Цей запит спочатку передається на сервер. Якщо в запиті зазначене доменне ім'я, фізичний сервер передасть запит DNS-серверу (програмі). DNS-сервер виділяє із запиту доменне ім'я – визначається відповідна IP-адреса. Далі сервер шукає IP-адресу в себе в базі даних або звертається до одному з кореневих серверів. Останній повертає вказівник на сервер відомої йому адреси наступного кроку транзиту й закінчує свою частину робіт. Тепер запит «перестрибує» в іншу мережу на сервер, що «знає», куди варто передати запит далі. У процесі аналізу адресації виконуються інші операції, зокрема, вибір найближчого шляху до того сервера, що при наявності декількох конкурентів-серверів забезпечить мінімум часу на перепрaвляння запиту. Такі «стрибки» запиту з мережі в мережу повторюються доти, поки не буде досягнутий адресат. Далі на запитуваній стороні формується TCP/IP-пакет даних і відправляється в протилежному порядку убік комп'ютера, що зробив запит. Якщо обсяг інформації, що транспортується, великий і не поміщається в один пакет, то дані розбиваються на ряд пакетів, які потім незалежно друг від друга переправляються на сторону комп'ютера, що зробив запит. Розмір одного пакета не перевершує 1500 байт. Розроблювачі Internet уважають, що розбивка інформації на пакети настільки малого розміру, з одного боку, дозволяє пересилати дані з оптимальною швидкістю, а з іншого боку, не створює «заторів» в інформаційних магістралях. Якщо при передачі якого-небудь пакета відбулася помилка, то операція передачі пакета виконується заново. Після одержання всіх пакетів вони збираються в єдине ціле, і операція обробки запиту вважається повністю виконаною. Якщо запит пішов від Web-

браузера, то користувач побачить на екрані монітора результати запиту.

## 17.6. Підключення до Internet

Існує два способи підключення комп'ютера до Internet - постійний і тимчасовий.

**Постійний спосіб** характеризується тим, що комп'ютер фізично підключений до якої-небудь локальної мережі, наприклад, мережі підприємства, що, у свою чергу, має постійне з'єднання з Internet.

Для того, щоб локальна мережа одержала доступ в Internet, необхідно вибрати *провайдера* – організацію, що робить платні Internet-послуги. Вибір провайдера для конкретної мережі провадиться виходячи з умов, у яких буде працювати ця мережа, і з умов, які включають ціну послуг і швидкість передачі даних, які ставить провайдер. Після укладання угоди між адміністрацією мережі й провайдером здійснюється підключення мережі до Internet.

Для постійного з'єднання комп'ютеру необхідні *мережна плата* й *мережний кабель* для зв'язування комп'ютера з локальною мережею організації. Якщо локальна мережа має зв'язок з Internet, то такий комп'ютер автоматично пов'язаний з Internet. Втім, це не виходить, що користувачі всіх комп'ютерів такої мережі можуть використовувати Internet постійно. Більшість комп'ютерів цілком може бути відключено адміністрацією мережі від зовнішнього миру з метою розвантаження власної мережі.

**Тимчасовий спосіб** складається в підключенні комп'ютера до Internet за допомогою *модему* – пристрою, що дозволяє приймати й передавати сигнали по телефонним проводам. Робота модему управляється спеціальним Ppp-протоколом. Цим способом користується зараз переважна більшість власників домашніх комп'ютерів.

Для модемного способу з'єднання власник комп'ютера повинен знайти підходящого провайдера й укласти угоду про умови роботи в Internet. Після цього власник одержує від провайдера ім'я користувача (user), пароль (password) і номер віддаленого телефону, по якому він буде підключати свій комп'ютер до Internet через власну телефонну консоль. Крім того, він може одержати повну інструкцію з настроювання свого комп'ютера на lnternet або попросити виконати цю роботу фахівців провайдера. Необхідно пам'ятати, що пропускна здатність телефонних ліній значно нижче кабельного з'єднання

Після вирішення питань підключення потрібно настроїти комп'ютер на зв'язок із сервером провайдера й установити на комп'ютері необхідне для роботи програмне забезпечення (наприклад, Web-браузер, поштовий клієнт, Ftp-клієнт – залежно від характеру робіт), після чого комп'ютер готовий до роботи в Internet. Для входу в lnternet власник комп'ютера повинен настроїти програму підключення до мережі провайдера.

## 17.7. Програми-браузери

Всесвітня комп'ютерна павутина WWW, споконвічно задумана як засіб обміну науковою інформацією, у цей час стала частиною повсякденного життя десятків мільйонів людей по всьому світлу.

Для успішної роботи в мережі Internet на комп'ютері необхідно встановити який-небудь Web-клієнт - програму *браузер,* за допомогою якої буде здійснюється подорож по WWW. Основне призначення браузера - перегляд web-сторінок і взагалі HTML документів, у тому числі й тих, які перебувають на локальному комп'ютері. Існує безліч таких програм. Найпоширенішими й багатофункціональними програмами-оглядачами (синонім: браузер), на сьогоднішній день, є різні версії Internet Explorer і Netscape фірми Microsoft., а також Mozilla і Opera.

Браузер Microsoft Internet Explorer входить до складу Windows, і це є однією із причин його неймовірної популярності.

Браузер Opera – порівняльна нова програма на ринку, до її переваг варто віднести невеликий обсяг, високу швидкість роботи й невисоких вимог до ресурсів роботи. Opera є комерційною програмою. Без реєстрації Opera працює в повному обсязі, але показує рекламний баннер.

Браузер Mozilla є вільно розповсюджуваним програмним продуктом. До складу MoziIla, крім браузера, поштового клієнта входять засіб обміну миттєвими повідомленнями, редактор Web сторінок і календар. До недоліків варто віднести більш високі вимоги до апаратних ресурсів ПК.

Загальним для всіх програм-браузерів є:

- пошук в Інтернет інформації, що цікавить нас;
- перегляд прийнятих з мережі web-сторінок;
- завантаження й збереження знайденої інформації;
- перегляд web-сторінок, з якими працювали останнім часом (місяць, тиждень назад, учора ...);
- створення бази даних (списку, бібліотеки) з адрес сайтів, які можуть бути корисні для роботи;
- робота з електронною поштою;
- робота з каналами, новинами;
- створення індивідуальних настроювань.

## 17.8. Пошук інформації в Internet

Пошук інформації в Інтернет є творчим процесом. На сьогоднішній день не існує строгої алгоритмічної процедури, завдяки якої можна гарантовано одержати позитивний результат.

### 17.8.1. Природна мова запитів

Більшість пошукових систем підтримують запити природною мовою. У рядку запиту можна просто написати слово або фразу, у якій позначено те, що Ви хочете знайти. Пошукова система самостійно проаналізує й обробить Вашу інформацію, а потім постарається знайти все, що ставиться до заданої теми.

Якщо запит не дав ніяких результатів, ви можете повторити пошук з використанням мови запитів. При вживанні мови запитів існує ряд правил, які треба знати й ураховувати при складанні запиту. Наприклад, можливостями розширеного пошуку, що є в більшості пошукових систем, правилами запису логічних виразів, які в різних пошукових машинах різні.

Ефективність пошукових запитів можна істотно підвищити, якщо використовувати при їхньому складанні логічні оператори, такі як AND (і), OR (або), NOT (не). Замість зв'язок (або в сполученні з ними можна використовувати символи &, |, +, -.)

### 17.8.2. Пошук по словосполученню

Пошук краще проводити по словосполученнях, чим по окремих словах. Трохи набраних у запиті слів, розділених пробілами, означають, що всі вони повинні входити в одна речення шуканого документа. Той же самий ефект зробить уживання символу &. Приклад: програми & мова & VBA & приклад. Пошукова машина задає пошук документів, що містять кожне з перерахованих слів.

Між словами можна поставити символ |, [або] щоб знайти документи, що містять кожне із цих слів (зручно використовувати при пошуку синонімів). Приклад: школа | гімназія | освітня установа | ліцей. Будуть знайдені документи, що містять хоча б одне з перерахованих слів.

### 17.8.3. Пошук по точній словоформі

Використання лапок " " і круглих скобок () дозволяє знайти ті документи, у яких ці слова йдуть строго підряд і в заданій формі (за замовчуванням слова перебувають у всіх формах). Знак мінус – дозволяє вивести слово із запиту.

Приклад: (Доступний відпочинок у Криму) – (агентство)

У запиті пошук буде вироблятися по набору ключових слів, крім всіх документів, де зустрічається сполучення «агентство».

### 17.8.4. Пошук по шаблоні

Якщо необхідно розширити запит, можна використовувати символ * для

позначення довільної частини слова або ? для позначення довільного символу. Наприклад, запит «школ*» дозволить знайти всі слова, які починаються з послідовності літер «школ». Зокрема, школа, школяр і т.д. На жаль, такий підхід не дозволяє розрізняти «сторонні» слова, що починаються із цієї ж послідовності. Наприклад, «кол*» де результатом будуть слова «колба», «колби», «колесо».

## 17.9. Пошукові системи Internet

Пошукові системи називають і пошуковими машинами, і пошуковими програмами й просто розвідувачі. Вони містять у собі три компоненти:

1) робот, що досліджує сайти й заносить сторінки в індекси системи;

2) індекс системи, де зберігаються перетворені особливим образом текстові складові всіх сторінок і текстових файлів, які відвідувалися і були проіндексоване роботом;

3) система пошуку – програма, що обробляє запит користувача, знаходить в індексі документи, що відповідають критеріям запиту, і виводить список знайдених документів у порядку убування релевантности.

У цей час більшість розвідувачів, із просто пошукових систем, перетворилося в портали, де крім пошуку надані різноманітні послуги, наприклад безкоштовна пошта, новини, пошук картинок, форуми й т.д.

Слід зазначити й те, що існує набір основних функцій, які характерні для всіх пошукових систем, але в той же час кожна з них має й свої особливості. Це може стосуватися й мови запитів, і способу індексування документів, характеру виводу результатів пошуку.

По способу організації пошуку пошукові сервера діляться на три типи:

- Каталоги (глобальні довідники);
- Пошукові машини;
- Мета-пошукові машини.

Останнім часом спостерігається зближення всіх типів пошукових систем. У каталогах, що містить тисячі адрес, часто реалізується пошук по природній мові запиту, що дозволяє не блукати по рубриках і знаходити адреси документів по темі, що цікавить. У пошукових машинах часто організують каталоги, рубрики в яких створюються вручну, а наповнення адресами виробляється автоматично. Всі пошукові сервера залежно від того як відбита в них мережа Інтернет діляться на глобальні й регіональні, які, у свою чергу, можуть займатися збором інформації із всіх напрямків або спеціалізуватися на який те вузькій тематиці. Глобальні пошукові сервери намагаються охопити всю мережу Інтернет, регіональні – тільки деяку її частину, пов'язану з конкретним географічним регіоном

### 17.9.1. Каталоги (глобальні довідники)

Каталоги в Інтернет є електронними аналогами систематичних каталогів звичайних бібліотек. Вони зберігають адреси й короткі анотації на мережні ресурси, мають ієрархічну структуру, у якій всі збережені адреси впорядковані по категоріях спеціально розробленого «дерева-рубрикатора» (початкової сторінки сервера). При пошуку інформації пошукові машини звертаються до однієї з її галузей (рубриці), що містить список адрес на мережні ресурси по обраній темі.

При використанні каталогів варто мати на увазі те, що їхнє наповнення в більшості випадків виробляється вручну. Наслідком цього є низька оперативність і швидкість нагромадження інформації, а так само її малий обсяг і принципова неможливість охопити весь Інтернет. Позитивним є те, що участь людини в роботі каталогу є гарантією якості: знайдена адреса, швидше за все, буде діючим, знайдений ресурс буде присвячений шуканій темі.

Найбільш популярними глобальними довідниками є: www.Yahoo.com; www.Aport.ru.

### 17.9.2. Пошукові машини

Пошукові машини в реальному житті аналогів не мають. Принцип їхньої дії полягає в періодичному скануванні деякого діапазону адрес із метою виявлення web-серверів і наступною індексацією їхнього вмісту. Отримані індексні бази використовуються надалі для організації повнотекстового пошуку, що здійснюється шляхом вводу ключових слів у спеціальний рядок запиту (поле вводу), яке звичайно розташоване на головній сторінці сервера. Результат пошуку представляє список адрес на сторінки, що містять зазначені в запиті слова.

При використанні пошукових машин варто мати на увазі, що інформація в їхні бази заноситься автоматично з деякою періодичністю. Наслідком цього є можлива невідповідність умісту знайдених документів темі пошуку й можлива наявність у результатах пошуку так званих «мертвих посилань», тобто посилань, що ведуть на вже не існуючого сервера або сторінки.

Популярні пошукові системи:

www.Google.ru; www.Yandex.ru; www.Rambler.ru; www.meta.ua.

### 17.9.3. Мета-пошукові машини

Мета-пошукові машини не мають власних баз даних. Робота їх зводиться до передачі введеного вами запиту відразу декільком пошуковим серверам з наступною обробкою відповідей і видачею результатів.

Приклад: www.MetaBot.ru - наймогутніший Російський мета-пошук.

## 17.10. Робота з електронною поштою

Електронна пошта (E-mail) є в сучасному діловому світі незамінним засобом зв'язку. Як і в інших видах комп'ютерного зв'язку, головним моментом у всій системі є взаємодія двох програм - поштового сервера й поштового клієнта.

**Поштовий сервер** звичайно встановлений у провайдера або в локальній мережі компанії адресата.

**Поштові клієнти** — це програми, за допомогою яких користувач може приймати й відправляти пошту. Таких програм існує досить багато, але принцип роботи у всіх однаковий.

Наприклад, програма Outlook Express є одним з модулів Web-браузера Internet Explorer. Вона призначена для роботи з електронною поштою й телеконференціями в мережі Internet. Програма встановлюється автоматично при інсталяції браузера й не вимагає додаткового програмного забезпечення. Усе, що потрібно для початку роботи з Outlook Express, це адреса поштової скриньки й вихід у мережу (локальну мережу організації або Internet).

Ця програма дозволяє використовувати кілька облікових записів для обслуговування декількох власників електронних поштових скриньок, робити переадресацію пошти, виконувати розмітку повідомлень, відправляти приєднані файли й багато чого іншого.

Однак все більшу популярність отримують зовнішні поштові сервери або **поштові служби**, такі як www.mail.ru; www.gmail.com; www.ukr.net і т.д. Вони надають сервіс безкоштовної електронної пошти. Абонентові досить зареєструватися, щоб одержати доступ до поштової скриньки.

У більшості випадків **поштова скринька** містить папки: *Вхідні, Відправлені, Вилучені, Чернетки, Кошик*, що відкриваються користувачем та можуть працювати з відповідними видами повідомлень (листів). Для відкриття своєї поштової скриньки абонент повинен зайти на поштовий сервер і ввести свою адресу електронної пошти.

**Адреса електронної пошти** (адреса поштової скриньки) – унікальний ідентифікатор, що визначає поштову скриньку, куди приходять повідомлення. У загальному випадку адреса електронної пошти виглядає так: **name@company**. Слово ліворуч від знака @ є ім'ям адресата, а права частина є доменним ім'ям сервера, на який приходять повідомлення адресату.

# КОНТРОЛЬНІ ПИТАННЯ ПО ДИСЦИПЛІНІ

1. Назначение операционной системы и ее функции. Основные характеристики ОС Windows.
2. Файловая система ОС Windows. Основные объекты файловой системы и их графическое представление.
3. Понятие файла в ОС Windows. Типы файлов.
4. Основные свойства файлов, атрибуты.
5. Программное обеспечение, его типы (прикладное и системное ПО).
6. Использование шаблонов при задании имен файлов.
7. Понятие папки в ОС Windows. Дерево папок. Путь (адрес) к файлу.
8. Понятие ярлыка в ОС Windows. Основные способы создания ярлыков.
9. Основные способы копирования и перемещения объектов в ОС Windows.
10. Использование буфера обмена для копирования и перемещения файлов ОС Windows.
11. Копирование и перемещение файлов путем перетаскивания мышью в ОС Windows.
12. Основные устройства хранения данных в ПК.
13. Единицы измерения информации.
14. Диски ОС Windows. Типы дисков.
15. Назначение текстового редактора MS Word и его интерфейс.
16. Режимы просмотра документов в MS Word.
17. Понятие шаблона в MS Word. Создание документа на основе шаблона, способы сохранения документа.
18. Настройка параметров страницы в MS Word.
19. Способы выделения фрагментов текста в MS Word.
20. Копирование и перемещение фрагментов текста в MS Word.
21. Основные правила ввода текста в MS Word.
22. Форматирование шрифта в MS Word.
23. Параметры форматирования абзацев в MS Word.
24. Понятие стиля оформления документа MS Word, копирование стиля.
25. Типы списков в MS Word и способы их создания.
26. Использование линейки для настройки параметров страницы и абзаца в MS Word.
27. Проверка правописания в документе MS Word, виды ошибок и способы их устранения.
28. Основные свойства графических объектов в MS Word, работа с рисунками.
29. Выделение графических объектов в MS Word, изменение порядка их следования, группировка.
30. Обтекание текстом графических объектов в MS Word, его способы и дополнительные параметры.
31. Нумерация страниц в документе MS Word, работа с колонтитулами.
32. Использование в MS Word автофигур, их свойства и форматирование.

33. Использование редактора формул в MS Word.
34. Создание таблиц в MS Word, их форматирование.
35. Способы выделения элементов таблиц в MS Word.
36. Сортировка данных в таблицах MS Word.
37. Выполнение вычислений в таблицах MS Word.
38. Назначение редактора электронных таблиц MS Excel, его интерфейс.
39. Структура книги MS Excel, работа с листами.
40. Ввод и редактирование данных в ячейки листа MS Excel.
41. Понятие диапазона ячеек в MS Excel, его адресация и способы выделения.
42. Форматирование содержимого ячеек в MS Excel.
43. Основные числовые форматы представления данных в MS Excel.
44. Копирование содержимого ячеек с помощью маркера заполнения в MS Excel.
45. Основные арифметические операции в MS Excel, приоритет выполнения операций, ввод формул.
46. Использование в MS Excel *Автосуммы* для вычисления суммы значений из диапазона ячеек.
47. Основные функции и категории функций в MS Excel. Использование *Мастера функций*.
48. Функции массива в MS Excel, назначение и способ ввода. Решение СЛАУ.
49. Работа со списками (базами данных) в MS Excel: сортировка, фильтрация, подведение итогов.
50. Создание сводных таблиц в MS Excel.
51. Используемые при вычислениях в MS Excel способы адресации ячеек.
52. Построение диаграмм в MS Excel.
53. Алгоритм, его свойства и способы представления.
54. Основные графические блоки, используемые при составлении блок-схемы алгоритма.
55. Основные типы вычислительных процессов.
56. Особенности линейного и разветвляющегося вычислительных процессов.
57. Этапы организации циклического вычислительного процесса.
58. Типы вычислительных процессов и их структура.
59. Цикл с предусловием, его структура и принцип работы.
60. Цикл с постусловием, его структура и принцип работы.
61. Цикл «Для», его структура и принцип работы.
62. Структура программы на VBA.
63. Описание переменных. Основные типы данных VBA.
64. Оператор присваивания. Запись арифметических выражений. Стандартные функции VBA.
65. Операторы ввода вывода данных VBA.
66. Условный оператор VBA.
67. Операторы циклов в VBA.
68. Основные понятия теории баз данных.

69. Основные объекты базы данных СУБД MS Access.
70. Основные принципы проектирования и нормализации базы данных.
71. Структура таблиц базы данных СУБД MS Access.
72. Основные типы данных полей таблиц СУБД MS Access.
73. Основные свойства полей таблиц СУБД MS Access.
74. Установление связей между таблицами базы данных СУБД MS Access.
75. Создание форм с помощью мастера в СУБД MS Access.
76. Создание отчетов с помощью мастера в СУБД MS Access.
77. Назначение и виды запросов в СУБД MS Access.
78. Создание запроса на выборку в режиме *Конструктора* в СУБД MS Access.
79. Вычисления в запросах на выборку в СУБД MS Access.
80. Параметрические запросы на выборку в СУБД MS Access.
81. Создание итоговых запросов на выборку в СУБД MS Access.
82. Создание перекрестных запросов в СУБД MS Access.

# СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Информатика. Базовый курс. 2-е издание / Под ред. С.В. Симоновича. СПБ.: Питер, 2004. – 640 с.

2. Кушнир А.Н. Новейшая энциклопедия Windows. – М.:Эксмо, 2009. – 960 с.

3. Пол Мак-Федрис. Microsoft Windows XP. Полное руководство. – М.:Диалектика-Вильямс, 2006. – 880 с.

4. Дженифер Кеттел и др. Microsoft Office 2003. Полное руководство. – М.:Эком, 2006. – 832 с.

5. Голицына О.Л., Попов И.И. Основы алгоритмизации и программирования: учебное пособие. – М: ФОРУМ, 2008. – 432 стр.

6. Слепцова Л.Д. Программирование на VBA в Microsoft Office 2010. Самоучитель. – М.:Диалектика-Вильямс, 2010. – 432 с.

7. Кузьменко В.Г. VBA 2003. Самоучитель. – М.:Бином-Пресс, 2010. – 432 с.

8. Методическое пособие. «Основы алгоритмизации»/ К.Н. Ефименко, Ю.Н. Добровольский, В.С. Ильяшенко – Донецк: ДонНТУ, 2005. – 75 с.

9. Методическое пособие. Программирование в среде Visual Basic for Application. Части 1 и 2./ В.В. Шамаев, К.Н. Ефименко и др.- Донецк: ДонНТУ, 2005. – 100 с.

10. Методичні вказівки і завдання до лабораторних робіт за курсом «Інформатика і основи програмування»/ К.М.Єфіменко, Ю.М. Добровольський. – Донецьк: ДВНЗ «ДонНТУ». – 2009. – 60 с.

11. Методичні вказівки і завдання до виконання лабораторних робіт і організації самостійної роботи студентів з дисципліни «Інформатика» (напрям підготовки 6.030508 «Фінанси і кредит»)/ К.М. Єфіменко, Ю.М. Добровольський. – Донецьк:ДВНЗ «ДонНТУ». –2012. –69 с.

# КОНСПЕКТ ЛЕКЦІЙ

з нормативної навчальної дисципліни циклу природничо-наукової та загальноекономічної підготовки

# ІНФОРМАТИКА

для студентів всіх форм навчання

Галузь знань: 0305 Економіка та підприємництво

Напрям підготовки: 6.030508 Фінанси і кредит

Спеціалізація: *Фінанси підприємства* (ФПР), *Банківська справа* (ФБС), *Фінансова аналітика* (ФА), *Оподаткування* (ФП)

**Укладачі** Єфіменко Костянтин Миколайович
Добровольський Юрій Миколайович
Кучер Тетяна Вікторівна

Підп. в печать 30.08.13 р. Формат 60х84 1/16. Папір KumLux.
Різографічна печать. Умов. печ. а. 10,70 Умов. кр.-отт. 10,75
Обл.-вид. а. 10,80 Тираж 50 екз. Замовлення № 20/10

ДВНЗ «Донецький національний технічний університет»
83001, м. Донецьк, вул. Артема, 58