



Міністерство освіти і науки України  
Сумський державний університет

## **КЕРУЮЧІ СИСТЕМИ**

Конспект лекцій

для студентів спеціальності 171 «Електроніка»  
усіх форм навчання

Суми  
Сумський державний університет  
2024

Керуючі системи: конспект лекцій / укладачі: Т. О. Протасова, О.А. Борисенко, О. В. Д'яченко. – Суми : Сумський державний університет, 2024. – 103 с.

Кафедра електроніки і комп'ютерної техніки

## ЗМІСТ

<b>Розділ I</b>	<b>Вступ в теорію керування</b>	4
<b>Лекція 1.</b>	Основні поняття в керуванні	4
<b>Лекція 2.</b>	Основні принципи керування	7
<b>Лекція 3.</b>	Узагальнена структура систем керування та її аналіз	25
<b>Лекція 4.</b>	Класифікація систем керування	31
<b>Лекція 5.</b>	Загальна характеристика керуючих автоматів	36
<b>Розділ II</b>	<b>Подання алгоритмів роботи керуючих автоматів</b>	39
<b>Лекція 6.</b>	Подання роботи керуючих автоматів граф-схемами алгоритмів	39
<b>Лекція 7.</b>	Подання роботи керуючих автоматів логічними схемами алгоритмів	52
<b>Лекція 8.</b>	Взаємні перетворення між граф-схемами і логічними схемами алгоритмів	54
<b>Лекція 9.</b>	Подання роботи керуючих автоматів за допомогою матричних схем алгоритмів	65
<b>Лекція 10.</b>	Побудова матричної схеми алгоритму за заданою граф-схемою алгоритму і навпаки	68
<b>Лекція 11.</b>	Перетворення логічної схеми алгоритму в матричну схему алгоритму і навпаки	71
<b>Лекція 12.</b>	Мінімізація матричної схеми алгоритму за кількістю логічних умов	74
<b>Розділ III</b>	<b>Керуючі автомати зі схемною логікою</b>	79
<b>Лекція 13.</b>	Керуючі автомати на комбінаційних схемах	79
<b>Розділ IV</b>	<b>Мікропрограмні автомати</b>	85
<b>Лекція 14.</b>	Найпростіші мікропрограмні автомати	85
<b>Лекція 15.</b>	Універсальні мікропрограмні автомати Уїлкса	94
	<b>Список літератури</b>	102

## Розділ I. Вступ в теорію керування

### Лекція 1. Основні поняття в керуванні

Зміна станів об'єкта, системи або процесу, що ведуть до досягнення поставленої мети, називається **керуванням**.

Матеріальний об'єкт будь-якої природи, на зміну станів якого спрямовані керуючі дії, називається **об'єктом керування**.

Такими об'єктами керування можуть бути автомобіль; хворий, що лікується; людина, яка навчається; продукція, що виробляється; економіка країни, військова операція; науковий експеримент.

Сукупність значень параметрів об'єкта керування називається його **станом**.

Наприклад, у хворого станом буде набір суб'єктивних та об'єктивних показників, таких, як самопочуття, температура, тиск, результати аналізу крові і т.п. Стан військової операції може визначатися кількістю втрат у себе та у противника, кількістю боєприпасів, моральним духом, зайнятою територією.

Будь-які зовнішні дії на об'єкт керування, що призводять до зміни його станів, називаються **керуючими діями**.

Так, ремонт автомобіля, який приводить його із несправного стану у справний, є керуючою дією. Керуючою дією є звільнення та прийняття на роботу персоналу та робітників на виробництві, змінення номенклатури продукції, що випускається, накази командира і т.п.

Керуюча дія передається через **сигнал**, що надходить на вхід об'єкта керування і впливає на його вихідну величину. Характер зміни керуючої дії за часом називається **законом керування**.

Стан, до якого прагнуть перевести об'єкт керування, називається **метою** керування.

Метою або завданням керування може бути, наприклад, одержання бажаного ефекту при лікуванні хворого; досягнення високої якості продукції, що випускається підприємством, або зниження її собівартості; підтримка заданого ступеня

матеріального добробуту суспільства; отримання переваги над противником в бою.

Саме керування, здійснюване за часом, є цілеспрямованим *процесом* з вироблення керуючих дій, спрямованих на змінення станів об'єкта керування, основаних на переробці інформації про нього.

Інформація є необхідним атрибутом процесу керування. Без неї керування неможливе. Так, процес керування в автомобілі складається з вироблення водієм керуючих дій (команд) на автомобіль. В результаті автомобіль змінює свої параметри – такі, наприклад, як напрямок руху або швидкість.

Для процесу керування необхідно знати і передбачити поведінку об'єкта керування при різних можливих зовнішніх діях на нього.

Зовнішні фактори, які діють на об'єкт керування та обумовлюють відхилення величин, що регулюються, від установлених, називаються *збурюючими діями (факторами)*.

Збурюючі фактори звичайно є випадковими величинами і завчасно не можуть бути надійно враховані. Так, водій автомобіля в достроковій перспективі заздалегідь не може спрогнозувати погоду, аварійні ситуації на дорозі, технічні несправності. Хоча деякі уявлення про всі ці фактори він, певна річ, має і в цілому враховує їх перед поїздкою.

Так чи інакше їх негативну дію при появі він намагається ліквідувати керуючими діями: ховається від непогоди, знижує швидкість, ремонтує автомобіль. При цьому здійснюється компенсація збурюючих факторів керуючими діями.

Керування завжди пов'язане з вибором однієї з можливих альтернатив. Якщо альтернатива єдина, то керування неможливе в принципі. Однак навіть якщо альтернатив багато, то далеко не завжди серед них знайдеться така, яка відповідає меті керування. Тому керування можливе тільки тоді, коли серед усіх можливих станів об'єкта керування знайдеться такий, за допомогою переходу в який можливе досягнення поставленої мети.

Властивість, яка характеризує можливість приведення об'єкта керування у заданий стан за допомогою керуючих дій називається його **керованістю**.

Об'єкт вважається керованим, якщо існує керування  $U(t)$ , яке забезпечує його переведення за час  $t$  із довільного початкового стану  $x_0$  в довільний стан  $x_i$  за кінцевий час.

Більш точно поняття керованості сформулював американський вчений Р.Е. Калман. Він ввів поняття **повної** та **часткової** керованості.

Керування, яке переводить за кінцевий час об'єкт керування з будь-якого заданого початкового стану  $x_0$  в будь-який інший стан  $x_i$ , у тому числі і кінцевий  $x_k$ , називається **повною керованістю** і **частковою**, якщо такої можливості не існує.

Таким чином, перш ніж розв'язувати задачу керування, необхідно з'ясувати, чи має вона розв'язок, тобто визначити, чи є в об'єкті керування потрібний стан, і якщо він є, то чи можливо його досягти за допомогою керуючих дій, які наявні.

Наприклад, якщо водій автомобіля хоче розігнати його до швидкості 140 км/год, то необхідно спочатку дізнатися, чи можливо за технічними характеристиками досягти цього взагалі, а потім з'ясувати, чи може саме цей автомобіль за своїм технічним станом здійснити це зокрема.

Для того, щоб керувати об'єктом керування з урахуванням різних збурюючих факторів, необхідно знати і передбачити поведінку об'єкта, тобто апріорно зберігати інформацію про нього. Ця інформація створює **інформаційну модель** поведінки об'єкта керування під впливом керуючих дій.

Основу складних інформаційних моделей створюють математичні моделі, які враховують не тільки дію на об'єкт в даний момент часу, а й в попередні. З метою побудови математичних моделей використовуються диференціальні рівняння, а також інші складні математичні підходи і методи.

Найбільш вивчені об'єкти керування, що мають технічну природу. Відповідно і теорія керування ними виявилася найбільш розробленою. Вона має назву **теорії автоматичного керування**.

## Лекція 2. Основні принципи керування

Керування ґрунтується на використанні низки принципів, найбільш простий з них – принцип керування *за збуренням*, який відомий також як *принцип розімкненого керування*, а також як *принцип Понселе-Чикалева*.

Цей принцип припускає *компенсування* впливу збурень в процесі вироблення керуючих дій. Величина та напрямок збурень *відомі завчасно*. Цей принцип ще інколи називають *компенсаційним керуванням*. Назва відображає той факт, що у цьому випадку в результаті керування компенсується вплив збурюючої величини.

Рисунок 1.1 ілюструє принцип керування за збуренням.

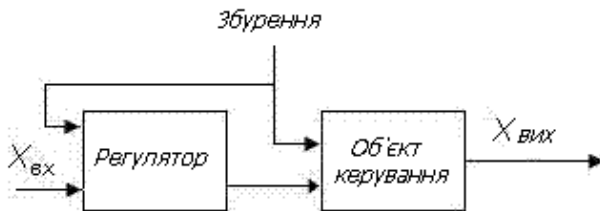


Рисунок 1.1 – Структурна схема принципу *розімкненого керування*

### *Застосування принципу розімкненого керування* -

Наприклад, можна вести автомобіль за відомим маршрутом за допомогою керуючого автомата, якщо до поїздки ввести в нього інформацію про необхідні для цієї мети керуючі дії. При цьому припускається, що завчасно відомі ті чи інші збурюючі фактори. Цей принцип також використовується при польоті на автопілоті, при керуванні по копіру в ткацьких верстатах, при пошиві за шаблоном.

Для цього принципу керування властиві певні *недоліки*. По-перше, впливів збурень на об'єкт може бути декілька, наприклад, зміна навантаження, температури середовища, рівня у резервуарі

тощо. Для забезпечення надійного керування **потрібно врахувати кожну** з величин збурення і для кожної будувати своє коло регулювання. Це практично здійснити неможливо, адже на роботу будь-якого об'єкта може впливати безліч причин. По-друге, для здійснення керування за збуренням необхідно повністю знати залежність реакції системи на збурюючу дію будь-якої величини й ввести цю залежність в алгоритм роботи регулятора. Тобто розробка регулятора передбачає попереднє вивчення поведінки системи при різних збуреннях, що не завжди можна здійснити з потрібним ступенем точності.

У той же час **керування за збуренням є найбільш простим і швидкодіючим**, а при ретельному вимірюванні збурень та врахуванні характеристик об'єкта керування – і достатньо точним.

### ***Історична довідка:***

**Жан-Віктор Понселе (Jean-Victor Poncelet):** 1 липня 1788 року, Мец, Франція – 22 грудня 1867 року, Париж.

Життя Понселе наочно демонструє той факт, що політ розуму, інтелекту неможливо стримати ніякими рамками. Математик знайде можливість займатися математикою у самих невідповідних для цього умовах.



Жан-Віктор Понселе (Jean-Victor Poncelet) народився у 1788 року бастардом — тобто незаконним сином досить багатого землевласника Клода Понселе. З'явитися на світ поза шлюбом на ті часи було зовсім нездорово, але як тільки Клод взяв як дружину мати Жан-Віктора, хлопчик відразу став законним сином і спадкоємцем.



Закінчивши звичайну школу, Понселе вступив до нещодавно створеної Політехнічної школи в Парижі. Завдання цього навчального закладу полягало у підготовці інженерних кадрів. Тут треба розуміти, що на початку XIX століття Франція була країною прикладників. А Політехнічна школа насправді більше була військовою — закінчивши її в 1810 році, Жан-Віктор спочатку отримав чин підпоручика, а потім і поручика (лейтенанта) інженерних військ.

Воював, був тяжко поранений і взятий у полон у 1812 році і потрапив до Саратова, де почав займатися математикою.

Понселе повернувся до Парижа, зробив ще багато великих відкриттів і навіть встиг кілька років покерувати Політехнічною школою. Досі існує легенда, що він змусив багатьох своїх підлеглих перейти на рахунки — нібито у Франції тоді не користувалися рахунками, а всі обчислення зазвичай проводилися «на папірці».

Понселе став членом багатьох зарубіжних академій наук, а його ім'я як одного з найвеличніших вчених Франції красується на таблиці на першому поверсі Ейфелевої вежі.

#### **Основні праці Ж.-В. Понселе:**

- (1822) *Traité des propriétés projectives des figures*
- (1826) *Cours de mécanique appliqué aux machines*
- (1829) *Introduction a la mécanique industrielle, physique ou expérimentale*
- (1838) *Théorie des effets mécaniques de la turbine Fourneyron*
- (1862/64) *Applications d'analyse et de géométrie.*

**Чиколев Володимир Миколайович:**

23 липня 1845 року – 22 лютого 1898 року – відомий російський інженер, конструктор та електротехнік, який вніс вагомий вклад у розвиток електричного освітлення та популяризацію наукових знань.

Він є автором проекту (разом з Р.Е. Классоном) центральної електростанції трифазного струму високої напруги.



Його зусиллями була створена диференційна лампа дугового світла, що дозволила вирішити питання розділення електричного струму. Крім цього, Чиколев В.М. працював над теорією прожекторів та створив перший у світі зразок швацької машинки з електродвигуном (1872 р.) Це був перший у світі електрифікований верстат.

Наступний принцип керування є основою всієї сучасної теорії керування. Цей принцип називається **принципом негативного зворотного зв'язку**, або просто **принципом зворотного зв'язку**. Вперше застосували цей принцип керування незалежно один від одного у 1765 р. І.І. Ползунов для регулювання рівня води і у 1784 р.- Дж. Уатт для центробіжного регулювання швидкості обертання двигунів. Тому принцип зворотного зв'язку ще називають **принципом Ползунова-Уатта**.

Використання принципу зворотного зв'язку збільшує або зменшує відхилення дійсних значень величин, що регулюються, від заданого значення, тому принцип зворотного зв'язку називається ще принципом керування **за відхиленням**.

Рисунок 1.2 ілюструє ідею принципу **негативного зворотного зв'язку**.

Принцип зворотного зв'язку використовується тоді, коли вплив збурюючих факторів значний і вони раніше невідомі, так що знехтувати ними або їх урахувати практично неможливо. Керування здійснюється в цьому випадку за кінцевим сумісним результатом впливу керуючих дій та збурюючих факторів. В

цьому як раз і полягає цінність принципу зворотного зв'язку, для використання якого нема необхідності у попередньому вимірюванні збурюючих факторів і дослідженні характеру потрібних керуючих дій. В результаті їх вплив враховується в процесі керування за знаком і величиною відхилення сигналу зворотного зв'язку. Цей сигнал дозволяє стежити за ефектом регулювання та усувати недо- і перерегулювання, збільшуючи тим самим точність керування. При цьому якщо сигнал зворотного зв'язку збільшується, то керуюча дія повинна зменшитись і, навпаки, якщо сигнал зворотного зв'язку зменшується, то керуюча дія збільшується. Це і є проявом *негативного* зворотного зв'язку.

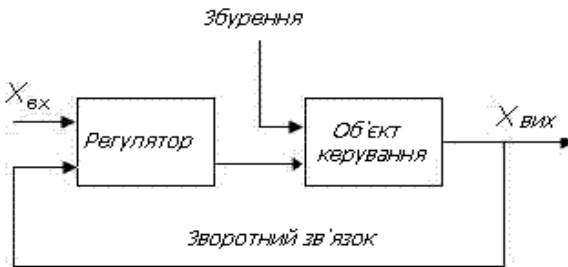


Рисунок 1.2 – Структурна схема принципу *негативного зворотного зв'язку*

Прикладом дії принципу зворотного зв'язку є керування автомобілем. Водій неперервно аналізує свої керуючі дії і вплив збурюючих факторів за сигналами зворотного зв'язку, які надходять до нього як через прилади, так і через вікна автомобіля. Реагуючи на ці сигнали, він виробляє ті чи інші керуючі дії, і автомобіль рухається.

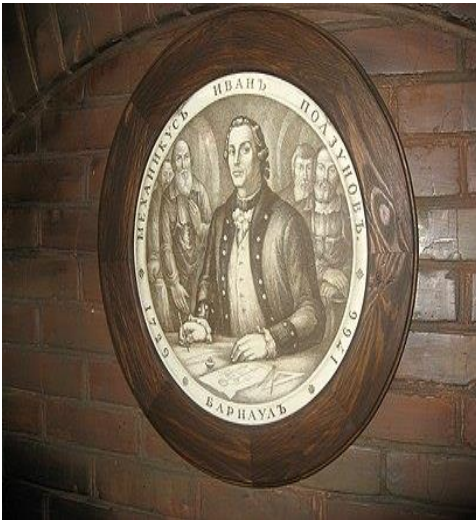
Іншим прикладом керування із зворотним зв'язком є керування бойовою операцією. Якщо залишити командира від результатів його приказів без зворотного зв'язку, то бій, швидше за все, буде програний, тому що централізоване керування боєм

буде втрачене, і його підлеглі будуть приймати керуючі рішення самостійно і, як правило, неузгоджено.

Недоліком керування із зворотним зв'язком є можливість втрати його стійкості. У цьому випадку регулюючі дії не встигають за зміненнями величини, що регулюється, і тоді замість послаблення впливу керуючих сигналів на вихід об'єкта керування за допомогою негативного зворотного зв'язку може здійснитися їх підсилення і як наслідок - втрата стійкості керування.

Наприклад, в аварійній ситуації при нестачі часу на обдумування ситуації водій автомобіля замість того, щоб натиснути на гальма, натискає на газ, і замість зменшення швидкість збільшується. Звичайно, протиаварійна стійкість автомобіля в цьому випадку різко знижується.

### *Історична довідка:*



Іван Іванович Ползунов (14 березня 1728, Єкатеринбург — 27 травня 1766, Барнаул) — російський винахідник-теплотехнік, винахідник першої в Росії паросилової установки. Його перша в Росії парова машина була також першим у світі двоциліндровим паровим двигуном, який вперше в історії не вимагав допоміжного гідравлічного приводу (здатного працювати без проточної води та водяного колеса).

Парова машина Ползунова — парова машина, створена навесні 1766 р. Іваном Ползуновим. Висотою вона була як триповерховий будинок, а її деякі деталі важили 2,5 тони. Працювала вона так: вода нагрівалась у котлі, склепаному із металевих листів і, перетворюючись у пару, надходила у два

триметрові циліндри. Поршні циліндрів тиснули на коромисла, що з'єднувались із міхами, які роздували полум'я в рудоплавильних печах, а також із водяними насосами-розподільниками. Також була передбачена автоматична подача в котел підігрітої води. Наявність двох поршнів забезпечувала неперервність роботи.



Джеймс Уатт (англ. James Watt; 19 (30) січня 1736 — 19 серпня 1819) — шотландський інженер, винахідник-механік. Удосконалив парову машину Ньюкомена. Винайшов універсальну парову машину подвійної дії. Роботи Уатта започаткували промислову революцію спочатку в Англії, а потім і в усьому світі. Увів першу одиницю потужності — кінську силу.

У 1782 році Британська асоціація інженерів вирішила присвоїти його прізвище одиниці потужності. Це був перший в історії техніки випадок присвоєння імені власного одиниці виміру. 29 травня 2009 року Банк Англії оголосив, що буде випущена банкнота в 50 £ із зображенням Уатта та Болтона. На даний момент банкнота випущена та перебуває в обігу.

Принцип *децентралізованого або директивного керування (від нім. Auftragstaktik)* використовується при керуванні великою кількістю складних об'єктів керування. Керування кожним з них здійснюється здебільшого самостійно. Цей принцип застосовують для прискорення процесу прийняття рішення. При чому право прийняття рішення за конкретними питаннями діяльності передається тому рівню керування, на якому ця діяльність здійснюється. Децентралізоване керування дозволяє покращити якість прийняття рішень, що приймаються як на вищому, так і на нижніх рівнях. При цьому здійснюється

робота з навчання та покращення професійних навичок штатного складу.

Граф Гельмут Карл Бєрнхард фон Мольтке, Мольтке Старший (нім. Helmuth Karl Bernhard Graf von Moltke; 26 жовтня 1800 Пархім, Німеччина — 24 квітня 1891 Берлін, Німеччина) — граф (1870), прусський і німецький воєначальник і військовий теоретик, генерал-фельдмаршал Пруссії (1871) і Російської імперії (1872), начальник Великого Генерального штабу Пруссії. Поряд з Бісмарком і Рооном вважається одним із засновників Німецької імперії.



*Eine günstige Situation kann nicht genutzt werden, wenn die Kommandeure auf Befehle warten. Jeder, vom höchsten Kommandeur bis zum jüngsten Soldaten, muss sich immer daran erinnern, dass Untätigkeit und Passivität schädlicher sind als falsche Handlungen.*  
– ідея та методологія принципу **децентралізованого керування** сформульована саме засновником принципу.

Гельмут Мольтке отримав всі можливі нагороди, всього нагороджений 55 орденами. Відомий своїм германофільством російський імператор Олександр II 30 серпня 1872 року призначив Гельмута Мольтке шефом 69-го піхотного Рязанського полку, яким він залишився до своєї смерті. У тому році отримав чин генерал-фельдмаршала Російської імперії. Крім того, був обраний почесним членом Санкт-Петербурзької академії наук 3 грудня 1871 за військово-теоретичні та військово-історичні праці.

Усього отримав п'ять найвищих нагород від Росії:

- орден Святого апостола Андрія Первозданного;
- орден Святого Георгія 2-го ступеня;

- орден Святого Олександра Невського;
- орден Білого Орла;
- орден Святої Ганни 1-го ступеня.

21 жовтня 1889 року фельдмаршал зробив два аудіозаписи на новому циліндричному фонографі Томаса Едісона. Перший містить вітальне послання Едісону та уривок із Фауста, другий – із Гамлета. Це єдині аудіозаписи голосу людини, яка народилася у XVIII столітті — Мольтке народився 1800 року (технічно останній рік цього сторіччя). За іронією долі, Хельмут фон Мольтке був відомий як *der große Schweiger* — «великий мовчун» за свою мовчазність.

Принцип *централізованого керування* застосовується у випадку, коли кількість об'єктів керування невелика і їх роботу необхідно жорстко узгоджувати між собою. У цьому випадку існує одне загальне централізоване завдання керування.

Це найголовніший принцип керування грошовою системою. Він передбачає наявність єдиного державного центру в особі центрального банку, міністерства фінансів або казначейства, що визначає основи організації грошового обігу і регулює його.

Зазвичай під централізацією прийнято розуміти концентрацію (зосередження) владних функцій прийняття управлінських рішень на верхньому ієрархічному рівні керівництва організації.

Централізація дозволяє більш ефективно здійснювати координацію і контроль діяльності структурних підрозділів щодо реалізації стратегічної політики організації в цілому.

Крім того, централізація керування дозволяє ефективно використовувати техніко-технологічні, матеріальні і кадрові ресурси, необхідні для вирішення цілей організації.

**До недоліків** відносяться:

- придушення творчої ініціативи персоналу в вирішенні виробничих завдань організації;
- зниження оперативного керування;

- зниження можливості адаптації персоналу до нових умов виробництва і роботи.

Принцип *екстремального керування* застосовується у випадку, коли підтримується режим керування, що характеризується максимально можливими деякими показниками якості керування із втратою інших.

Цей принцип розглянемо на прикладі **екстремального програмування** або *XP, eXtreme Programming* - гнучка методологія розробки програмного забезпечення. Автор *XP* не придумав нічого нового, а взяв кращі практики гнучкої розробки і посилив до максимуму. Тому програмування і називається екстремальним.

Автор методики – американський розробник Кент Бек. В кінці 90-х років він керував проектом *Chrysler Comprehensive Compensation System* і там вперше застосував практики екстремального програмування. Свій досвід та створену концепцію він описав в книзі *Extreme Programming Explained*, що була опублікована в 1999 році. За нею були надруковані і інші книги, в яких докладно були описані практики *XP*. До становлення методології причетні також *Уорд Каннингем, Мартин Фаулер* та інші.

Хто використовує, наприклад: *Pivotal Software, Inc*, яка розробляє ПЗ для бізнес-аналізу на основі *big data* і надає консультаційні послуги.

Продуктами *Pivotal* користуються корпорації *Ford, Mercedes, BMW, GAP, Humana*, великі банки, державні установи, страхові компанії.

**Мета методики XP** - впоратися з постійно мінливими вимогами до програмного продукту та скоротити вартість неочікуваних змін і підвищити якість розробки.

Тому *XP* добре підходить для складних і невизначених проектів. У традиційних методах розробки вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану. *XP* використовується для



скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Методологія XP будується навколо чотирьох процесів: кодування, тестування, дизайну і слухання. Крім того, екстремальне програмування має цінності: простоту, комунікацію, зворотний зв'язок, сміливість і повагу.

Графік (схема потоку робіт в XP- див. рис.1.3) показує, як правила екстремального програмування працюють разом. Клієнти люблять бути партнерами в процесі розроблення програмного забезпечення і активно сприяти незалежно від рівня досвіду, а менеджери мають бути зосередженими на зв'язках і відношеннях. У результаті ми отримуємо метод, який потенційно володіє високою адаптацією до серйозно і часто мінливих вимог до проекту, але в той же час не позбавлений ряду принципових недоліків і дуже високого ступеня залежності від людського фактору.

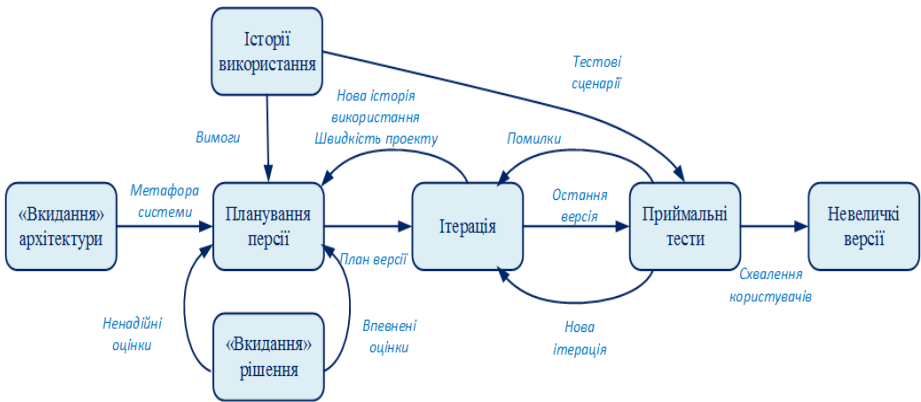


Рисунок 1.3 - Схема потоку робіт в XP

Дванадцять основних прийомів екстремального програмування (Кругова схема-ілюстрація та Стрілкова діаграма-ілюстрація прийомів екстремального програмування наведені відповідно на рисунках 1.4 та 1.5) можуть бути об'єднані в чотири групи:

**1) Короткий цикл зворотного зв'язку (Fine-scale feedback):**

Розробка через тестування (Test-driven development)

Гра в планування (Planning game) - швидко сформувати приблизний план роботи і постійно оновлювати його в міру того, як умови задачі стають все більш чіткими

Замовник завжди поруч (Whole team, Onsite customer) - XP стверджує, що замовник повинен бути весь час на зв'язку і доступний для питань.

Парне програмування (Pair programming) - передбачає, що весь код створюється парами програмістів, які працюють за одним комп'ютером. Один з них працює безпосередньо з текстом програми, інший переглядає його роботу і стежить за загальною картиною того, що відбувається.

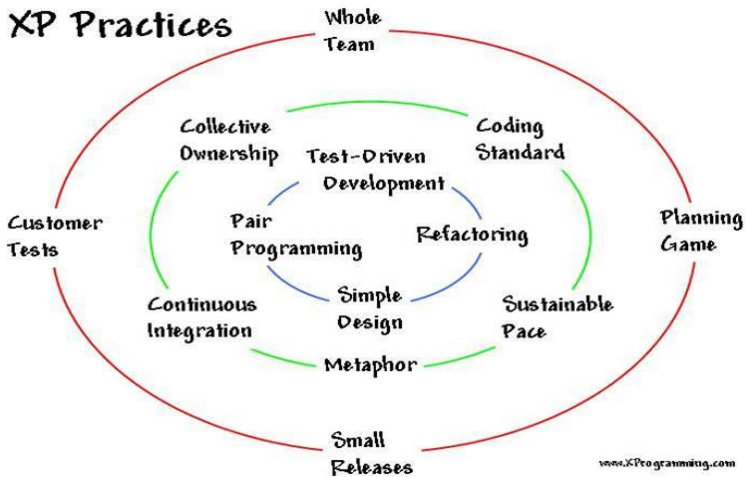


Рисунок 1.4 – Кругова схема-ілюстрація прийомів екстремального програмування

**2) Безперервний, а не пакетний процес:**

Безперервна інтеграція (Continuous integration) - В XP інтеграція коду всієї системи виконується кілька разів на день,

після того, як розробники переконалися в тому, що всі тести модулів коректно спрацьовують.

Рефакторинг (Design improvement, Refactoring) - XP має на увазі, що одного разу написаний код в процесі роботи над проектом майже напевно буде неодноразово перероблений.

Тести модулів дозволяють розробникам переконатися в тому, що їх код працює коректно. Вони також допомагають іншим розробникам зрозуміти, навіщо потрібен той чи інший фрагмент коду та як він функціонує. Тести модулів також дозволяють розробнику без будь-яких побоювань виконувати рефакторинг. Розробники XP безжально переробляють написаний раніше код для того, щоб поліпшити його

Часті невеликі релізи (Small releases) - версії (releases) продукту повинні надходити в експлуатацію якомога частіше. Робота над кожною версією повинна займати якомога менше часу. При цьому кожна версія має бути достатньо осмисленою з точки зору корисності для бізнесу.

### **3) Розуміння, що розділяється всіма:**

Простота (Simple design) - XP виходить з того, що в процесі роботи умови задачі можуть неодноразово змінитися, а значить, що розробляється продукт не слід проектувати завчасно цілком і повністю. Проектування повинно виконуватися невеликими етапами, з урахуванням постійно змінюються вимог. У кожен момент часу слід намагатися використовувати найбільш простий дизайн, який підходить для вирішення поточної задачі, і міняти його в міру того, як умови задачі змінюються.

Метафора системи (System metaphor) - це уявлення про компоненти системи і їх взаємозв'язках між собою. Розробникам необхідно проводити аналіз архітектури програмного забезпечення для того, щоб зрозуміти, в якому місці системи необхідно додати нову функціональність, і з чим буде взаємодіяти новий компонент.

Коллективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership)-означає, що кожен член команди несе відповідальність за весь вихідний код. Таким чином, кожен має

право вносити зміни в будь-яку ділянку програми. Парне програмування підтримує цю практику: працюючи в різних парах, всі програмісти знайомляться з усіма частинами коду системи. Важлива перевага колективного володіння кодом в тому, що воно прискорює процес розробки, оскільки при появі помилки її може усунути будь-який програміст.

Стандарт кодування (Coding standard or Coding conventions) - в рамках XP необхідно домогтися того, щоб було складно зрозуміти, хто є автором тієї чи іншої ділянки коду, - вся команда працює уніфіковано, як одна людина. Команда повинна сформувати набір правил, а потім кожен член команди повинен слідувати цим правилам в процесі кодування. Перелік правил не повинен бути вичерпним або занадто об'ємним. Завдання полягає в тому, щоб сформулювати загальні вказівки, завдяки яким код стане зрозумілим для кожного з членів команди.

#### 4) Соціальна захищеність програміста (Programmer welfare):

40-годинний робочий тиждень (Sustainable pace, Forty-hour week).

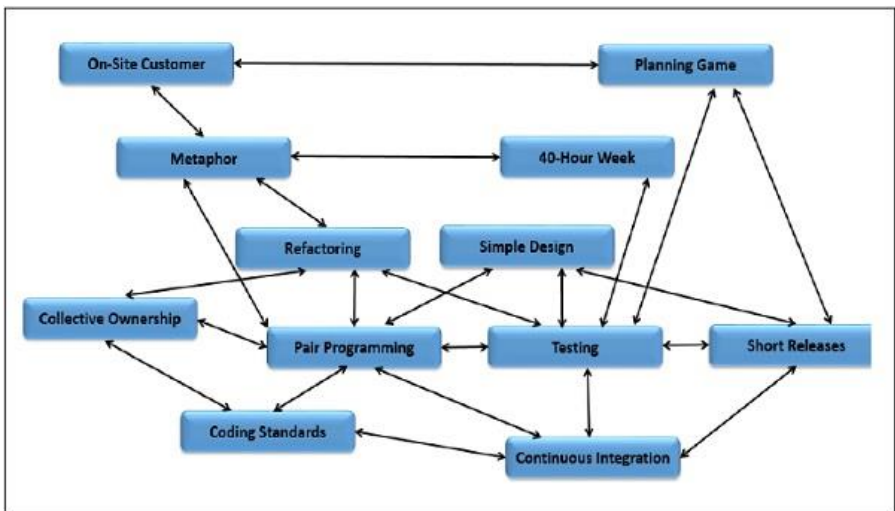


Рисунок 1.5 – Стрілкова діаграма-ілюстрація прийомів екстремального програмування

### ***Переваги екстремального програмування:***

- Замовник отримує саме той продукт, який йому потрібен, навіть якщо на початку розробки сам точно не уявляє його кінцевий вигляд.
- Команда швидко вносить зміни в код і додає нову функціональність за рахунок простого дизайну коду, частого планування і релізів.
- Код завжди працює за рахунок постійного тестування і безперервної інтеграції.
- Команда легко підтримує код, тому що він написаний за єдиним стандартом.
- Швидкий темп розробки за рахунок парного програмування.
- Висока якість коду.
- Знижуються ризики, пов'язані з розробкою, тому що відповідальність за проект розподіляється рівномірно.

Витрати на розробку нижче, тому що команда орієнтована на код, а не на документацію.

### ***Недоліки екстремального програмування:***

- Успіх проекту залежить від залучення замовника, якої не так просто домогтися важко передбачити витрати часу на проект, тому що на початку ніхто не знає повного списку вимог.
- Успіх ХР сильно залежить від рівня програмістів, методологія працює тільки з senior фахівцями.
- Менеджмент негативно відноситься до парного програмування, не розуміючи, чому він повинен оплачувати двох програмістів замість одного.
- Регулярні зустрічі з програмістами дорого обходяться замовникам.
- Вимагає занадто сильних культурних змін.
- Через нестачу структури і документації не підходить для великих проектів.

**Цікавий матеріал про екстремальне програмування:**

[http://www.info-system.ru/article/article\\_extreme\\_prog.html](http://www.info-system.ru/article/article_extreme_prog.html)

<https://ppt-online.org/665110>

<https://worksection.com/blog/extreme-programming.html>

<https://worksection.com/ua/blog/extreme-programming.html>

[https://wiki.cuspu.edu.ua/index.php\(XP\)](https://wiki.cuspu.edu.ua/index.php(XP))

[https://elearning.sumdu.edu.ua/free\\_content/lectured](https://elearning.sumdu.edu.ua/free_content/lectured)

Принцип **адаптивного керування** використовується у випадку, коли необхідно процес керування пристосовувати до змін збурюючих факторів і змін станів об'єкта керування з метою поліпшення якості керування.

**Адаптивне керування** – сукупність методів теорії керування, що дозволяють синтезувати системи керування, які мають можливість змінювати параметри регулятора або структуру регулятора в залежності від змін параметрів об'єкта керування або збурюючих факторів, що впливають на об'єкт керування. Системи, що побудовані за таким принципом, називають адаптивними. Адаптивне керування досить поширене в багатьох застосуваннях теорії керування.

**За характером змін в керуючому пристрої** адаптивні системи поділяються на дві великі групи:

- Системи, що самоналаштовуються – змінюються лише значення параметрів регулятора;
- Системи, що самоорганізуються – змінюється структура самого регулятора.

**За способом вивчення об'єкта** системи поділяють на:

- Пошукові;
- Безпошукові.

У першій групі особливо відомі екстремальні системи, метою керування яких є підтримка системи в точці екстремуму статистичних характеристик об'єкта.

**За способом досягнення ефекту самоналаштування** системи поділяються на:

- Системи з сигнальною (пасивною);

- Системи з активною (параметричною) адаптацією;
- Комбіновані системи (якщо система об'єднує в собі обидва види адаптації).

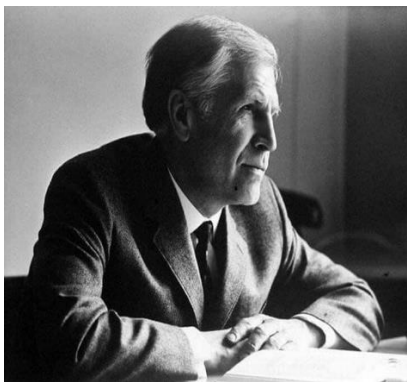
Принцип *випадкового керування* виникає у неповноті визначених ситуаціях. Керування вибирається з деяких можливих стратегій, як правило, випадково.

При вирішенні таких завдань керування доводиться мати справу з ситуаціями, коли знаходження оптимального рішення ускладнюється необхідністю одночасного врахування багатьох невизначеностей і відповідних сценаріїв їх випадкового впливу на роботу системи.

Принцип *програмного керування* використовується у випадку, коли необхідно, щоб стани об'єкта керування змінювались згідно *заздалегідь* заданим законом. У цьому випадку між станами об'єкта керування і керуючими впливами встановлюється функціональний зв'язок. Закон зміни керуючих впливів встановлюється при цьому *априорно*.

Принцип *стежачого керування* працює аналогічно до принципу програмного керування з тією відмінністю, що зміни керуючих впливів заздалегідь не встановлені і є *випадковими*.

Існують й інші, більш часткові принципи керування, наприклад *динамічне програмування* або за *принципом максимуму Понтрягіна*.



### ***Лев Семенович Понтрягін***

(3 вересня 1908 р., Москва - 3 травня 1988 р., Москва) – радянський математик, один з найвидатніших математиків ХХ століття, академік АН СРСР (1958; член-кореспондент з 1939). Герой Соціалістичної Праці (1969). Лауреат Ленінської премії (1962), Сталінської премії другого ступеня (1941) та Державної премії СРСР (1975).

***Лев Семенович Понтрягін*** зробив значний внесок в алгебраїчну та диференціальну топологію, теорію коливань, варіаційне числення, теорію управління. У теорії управління Понтрягін - засновник математичної теорії оптимальних процесів, в основі якої лежить принцип максимуму Понтрягіна; має фундаментальні результати з диференціальних ігор. Роботи школи Понтрягіна справили великий вплив на розвиток теорії управління та варіаційного числення в усьому світі.

При вирішенні варіаційних задач класичними методами труднощі виникають у тих випадках, коли відшукувані управляючі дії не належать до класу безперервних функцій або коли на змінні задачі накладені обмеження типу нерівностей. Такого роду завдання носять назву завдань про швидкодію і вирішення їх можна отримати, використовуючи принцип максимуму Понтрягіна.

У вирішенні завдань ракетодинаміки зі складними обмеженнями принцип максимуму Понтрягіна завоював особливу популярність, з чим пов'язаний великий прогрес, досягнутий у всьому світі при вирішенні практичних завдань ракетодинаміки.



### Лекція 3. Узагальнена структура систем керування та її аналіз

Розглянуті принципи керування вирішуються за допомогою *систем керування*.

Сукупність об'єктів керування та технічних засобів дії на них називається *системою керування*.

Технічні засоби, призначені для цілеспрямованої дії на об'єкт керування, називаються *керуючими системами*.

Так, наприклад, водій та автомобіль сумісно утворюють систему керування, в якій об'єктом керування є автомобіль, а керуючою системою - водій. Процес керування складається при цьому з вироблення водієм керуючих дій (команд) на органи керування автомобіля - кермо, коробку передач, газ, гальма, прилади освітлення, опалення і т.д.

Керуюча система та об'єкт керування з'єднуються через їх виходи і входи в єдину *систему керування* (див. рис. 1.6). Її структура задає всі основні параметри системи керування, такі, як швидкодія, ємність пам'яті, надійність, живучість. Вона складається у загальному випадку із керуючих засобів, елементів порівняння, інформаційної системи, системи збирання і передавання даних, виконавчих та регулюючих органів, засобів відображення інформації та пульта керування.

Одна й та сама за своїми функціями система керування може бути реалізована за допомогою різних окремих структур, що відрізняються своїми характеристиками, наприклад, надійністю і швидкодією. Пошук серед цих структур найбільш адекватної для того чи іншого завдання керування є завдання вибору найбільш ефективної структури системи керування.

Завданням керуючих засобів є вироблення керуючих дій, що надходять через виконавчі та регулюючі органи на об'єкт керування. Ці засоби є комплексом різних апаратних і програмних засобів у вигляді різного типу перетворювачів інформації як аналогових, так і цифрових, у тому числі й ЕЦОМ, які виробляють дії на об'єкт керування за певним законом.

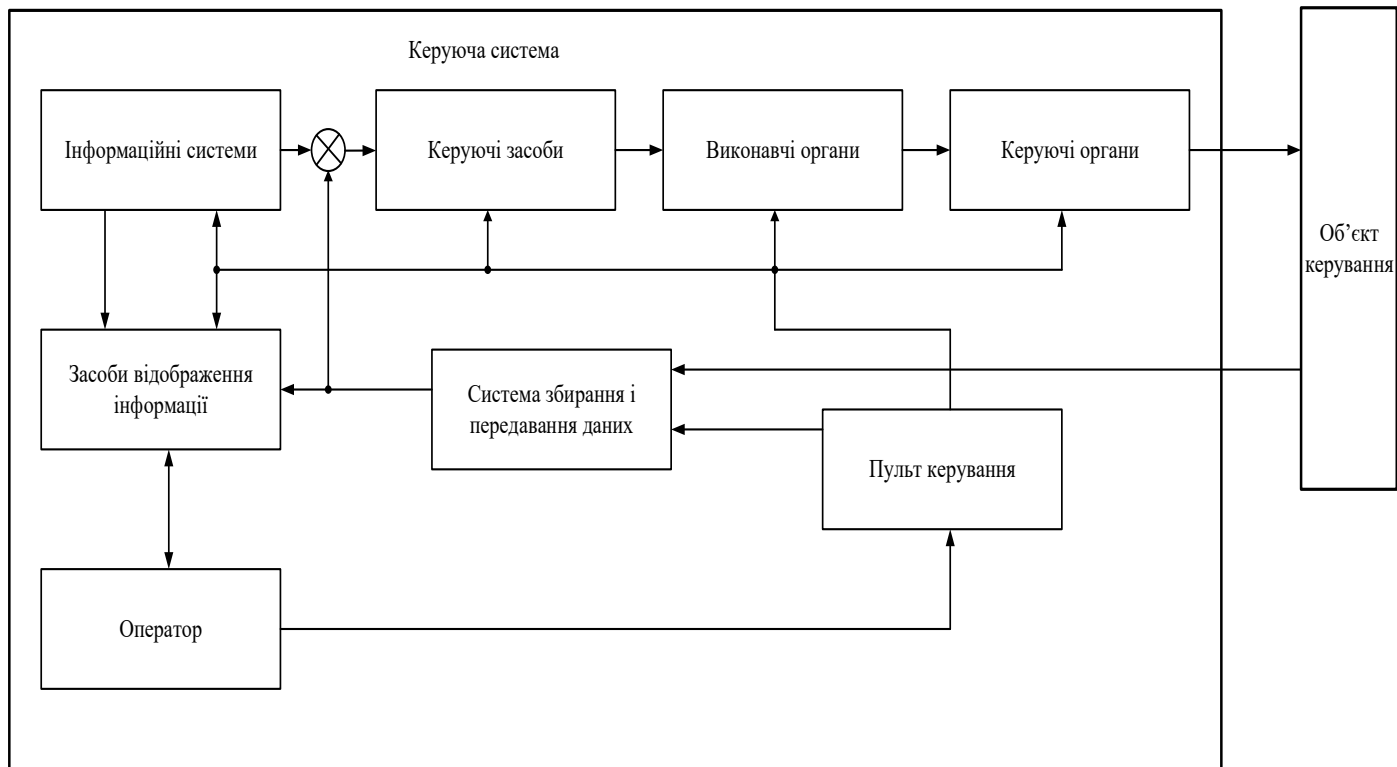


Рисунок 1.6 – Структура системи керування

Ці дії надходять у виконавчі органи, які є електричними, електропневматичними та електрогідравлічними сервомеханізмами, залучають до роботи регулюючі органи.

Регулюючі органи являють собою різні засувки, заслінки, клапани, помпи і т.д. Вони змінюють, наприклад, витрату палива, температуру, тиск пари, величину струму, навантаження і кількість обертів двигуна.

Наприклад, в автомобілі регулюючими органами будуть кермо, педаль газу, гальма, а виконавчими – руки і ноги водія.

Інформаційна система апріорно зберігає всю необхідну інформацію про об'єкт керування, а також мету, критерії та програми керування, утворюючи тим самим інформаційну модель об'єкта керування, з метою підвищення ефективності прийняття керуючих рішень у цю модель за необхідності вводиться також математична модель об'єкта керування.

Очевидно, що більш складну інформаційну модель має людина. Вона утворює її протягом усього свого життя, починаючи з раннього дитинства. У цій моделі містяться взаємопов'язані блоки за різними видами його професійної та суспільної діяльності. Її особливостями є гнучкість і великі адаптаційні здібності до навколишніх обставин, з також здібність до самонавчання. Тому багато систем керування не можуть обійтись без участі в них людини.

Наприклад, це стосується автомобіля, тому що для того щоб його нести, необхідно достатньою мірою володіти інформацією про будову автомобіля, його технічний стан, правила дорожнього руху, знати мету і маршрут поїздки, стан шляху, погодні умови і багато інших факторів. Водій повинен гнучко, в залежності від обставин, використовувати свої знання в процесі руху і тільки тоді може бути одержаний задовільний результат від його поїздки.

Система збирання і передавання даних призначена для збирання і передавання інформації про параметри об'єкта керування. Ці дані знімаються у вигляді сигналів з датчиків, що знаходяться безпосередньо на об'єкті керування. За необхідності

сигнали підсилюються і за допомогою системи передавання даних надходять до керуючої системи на елемент порівняння.

Так, в автомобілі системи збирання і передавання даних розв'язують задачі інформування водія про кількість бензину в баці, швидкість руху, температуру двигуна і т.д.

На елементі порівняння отримані сигнали порівнюються з сигналами, що надходять від інформаційної системи. В результаті визначається величина розходження - відхилення реальних параметрів об'єкта відносно необхідних їх значень.

За допомогою керуючих засобів це розходження усувається шляхом вироблення ними відповідних дій на виконавчі пристрої, які, в свою чергу, діють на регулюючі органи.

Останні змінюють параметри об'єкта керування таким чином, щоб існуюче розходження знизити до мінімуму, а в ідеалі - до нуля, в чому і полягають мета і завдання будь-якого керування.

Система відображення інформації необхідна у тих випадках, коли в контурі керування перебуває людина. Вона розв'язує задачу подання їй інформації у зручному вигляді, наприклад, у формі графіків, таблиць, мнемограм і т.д.

Пульт керування призначений для організації дії на хід керування оператором. Він може через цей пульт впливати у самому загальному випадку на будь-який блок керуючої системи, вносячи в його роботу необхідні корективи.

Входи і виходи керуючої системи та об'єкта керування призначені також для організації прямих і зворотних зв'язків у системі керування, узагальнена структура якої наведена на рис. 1.7.

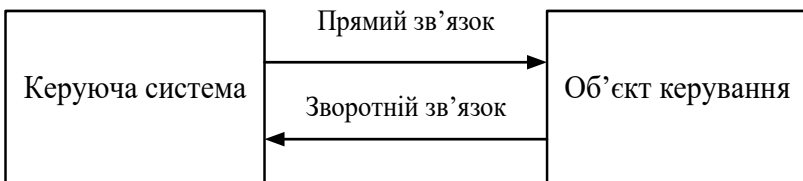


Рисунок 1.7 – Узагальнена структура системи керування

Зв'язок від керуючої системи до об'єкта керування називається *прямим*, а від об'єкта керування до керуючої системи - *зворотним*.

За допомогою прямого зв'язку здійснюється процес керування об'єктом - змінення його станів у бажаному напрямку, а за допомогою зворотного зв'язку передається інформація про реальний стан об'єкта керування - керуючої системи. Порівняння цього стану з бажаним визначає величину розходження і, отже, визначає подальші дії керуючої системи щодо змінення станів об'єкта керування.

Звичайно процес керування здійснюється таким чином: від об'єкта керування до керуючої системи передається інформація про значення параметрів об'єкта керування. Керуюча система порівнює їх з потрібними значеннями параметрів, що зберігаються у її пам'яті, і визначає величину їх розходження. Потім вибирається засіб усунення цього розходження, який потім реалізується керуючою системою.

Так, водій, дивлячись на приладний щит автомобіля, визначає реальну його швидкість і порівнює її з потрібною. Потім, враховуючи стан шляху, погодні умови та інші фактори ризику, розганяє свій автомобіль до потрібної швидкості.

Системи керування в реальних обставинах працюють в умовах *зовнішніх* і *внутрішніх* збурень.

Під зовнішніми збуреннями, або завалами, розуміють збурення, що надходять із зовнішнього середовища, а під внутрішніми - завади, або відмови, що виникають у самій системі керування.

Тому важливою проблемою, яка вирішується при проектуванні системи керування, є підвищення її *безпеки*, *надійності*, *завадостійкості* та *живучості*.

Сукупність властивостей системи керування, яка дозволяє уникнути аварії, називається її *безпекою*.

Здібність системи керування виконувати свої функції протягом заданого відрізка часу, називається її *надійністю*.

Здібність системи керування виконувати свої функції в умовах завад називається *завадостійкістю*.

Можливість виконання системою керування своїх основних функцій при відмові частини обладнання називається *живучістю*.

Сучасні системи керування так чи інакше вирішують ці завдання, однак вимоги до них неперервно зростають, і тому боротьба із зовнішніми і внутрішніми збуреннями в системах керування є на сьогодні актуальним завданням.

## Лекція 4. Класифікація систем керування

Для правильного та ефективного використання систем керування на практиці потрібна їх класифікація, в рамках якої можна дослідити особливості тих чи інших їх структур. В основу класифікації систем керування покладені різні ознаки, наприклад, види сигналів, принципи і типи керування. Розглянемо більш поширені класифікаційні ознаки і відповідні класи систем керування.

В першу чергу системи керування поділяються за ознакою типу об'єкта керування, яким вони керують.

Основі об'єкти керування можна поділити на *біологічні*, *технічні* та *соціально-економічні*. Відповідно і системи керування поділяються за цими ознаками.

До першого класу відносять усі живі істоти та їх органи, наприклад, клітини, віруси, бактерії, тварини, люди.

До другого класу відносять наукові, технологічні процеси і виробництва, з також різні пристрої та обладнання.

Третій клас складають соціально-економічні утворення - установи, армія, держава і т.д.

Для кожного класу наведених об'єктів керування повинні розроблюватись свої системи керування, які відповідно поділяють на *біологічні*, *технічні* та *соціально-економічні*.

За формою сигналів, що використовуються системами керування, вони поділяються на *аналогові* та *цифрові*.

У недавньому минулому найбільш поширеними були аналогові системи керування. Їх особливістю є використання аналогових величин як носив інформації: напруги, струму, частоти, фази і т.д. Перевагою аналогових систем керування є неперевершена швидкодія і відносно недорого і компактна їх реалізація. Однак точність і надійність роботи цих систем були недостатніми для все більш нових їх застосувань на практиці, тому відносно недавно були розроблені цифрові системи керування, основу роботи яких складали цифрова обробка сигналів і відповідно цифрове подання сигналів. Їх головна перевага - це висока точність роботи. Поряд з цим значно зросли

їх надійність, завадостійкість і живучість. Крім того, цифрові системи керування мають небачену для аналогових систем гнучкість та універсальність. Швидкодія цифрових систем керування хоча й менша від швидкодії аналогових, але для більшості практичних випадків виявляється достатньою. В результаті цифрові системи на сьогодні зайняли монопольне положення в усіх сферах використання систем керування.

Однак обійтись без аналогових елементів цифрові системи та пристрої в принципі не можуть, тому правильно було б говорити не про цифрові системи керування, а про цифрові з елементами аналогових. Крім того, в багатьох сферах практичного застосування, де вимоги до точності не настільки високі і необхідна висока швидкодія або мала вартість, достатньо ефективно використовуються аналогові системи керування.

За видом зв'язку системи керування поділяються на системи з *прямим* і *зворотним* зв'язком. У першому випадку інформація передається тільки від керуючої системи до об'єкта керування, а в другому - існує також передавання інформації в протилежному напрямку від об'єкта керування до керуючої системи.

Системи керування з прямим зв'язком називаються ще системами з *розімкненим* зв'язком, а із зворотним - із *замкненим*.

Системи керування з прямим або розімкненим зв'язком використовуються у відносно простих випадках, коли вплив збурюючих факторів незначний або їх можна передбачити, а закон керування заздалегідь невідомий.

У більш складних, найбільш поширених випадках, використовуються системи керування із зворотним зв'язком, тому що вони не потребують повної інформації про збурюючі дії та усі характеристики об'єкта керування.

Однак наявність зворотного зв'язку може призвести до зниження, а то і до втрати стійкості системи керування. В результаті погіршиться якість керування або система взагалі може припинити свою роботу.

Наступна ознака для класифікації систем керування це рівень автоматизації об'єкта керування.



Використання автоматичних пристроїв і систем для виконання *функцій керування* називається *автоматизацією*.

Ефект автоматизації виявляється насамперед у підвищенні продуктивності праці та якості продукції, а також у заміні людини автоматами у небезпечних і важкодоступних місцях, таких, як шкідливі хімічні виробництва, ядерні двигуни і реактори, космічні апарати і т.д.

При автоматизації основні процеси одержання енергії, матеріалів або інформації здійснюються *автоматично*, тобто за програмою без втручання людини.

Розрізняють такі три види автоматизації:

1. Часткова, коли автоматизуються не пов'язані один з одним механізми та устаткування.

2. Комплексна, коли автоматизуються як основні, так і допоміжні операції.

3. Повна - у випадку автоматизації усіх агрегатів та устаткувань, що беруть участь у робочому процесі.

У відповідності до видів автоматизації системи керування поділяються на системи з *частковою, комплексною і повною* автоматизацією.

Системи з частковою і комплексною автоматизацією мають назву *автоматизованих*. В них як учасник процесу керування обов'язково присутня людина.

Людино-машинні системи, засновані на використанні економіко-математичних методів і технічних засобів для розв'язування різних задач у виробництві, науці, техніці, освіті, військовій справі, проектуванні, плануванні, називаються *автоматизованими системами керування*.

Передумовою створення автоматизованих систем керування є можливість автоматизації інформаційних процесів на основі цифрових ЕОМ. Основними функціями автоматизованих систем керування є збирання, передавання, зберігання і оброблення первинних даних, формування документів для управлінського персоналу, видавання довідкової інформації, вироблення рекомендацій щодо керування.

Автоматизовані системи керування в залежності від об'єкта керування поділяються на ряд підкласів. З них найбільш відомі *автоматизовані системи керування підприємством (АСКП)* і *автоматизовані системи керування технологічним процесом (АСКТП)*.

АСКП є системами керування виробничо-господарчою діяльністю підприємства, що базується на комплексному використанні економіко-математичних методів та сучасних засобів оброблення інформації. Необхідність створення і втілення АСКП пов'язана з великою кількістю об'єктів керування, масштабністю виробництва і високою їх взаємозалежністю. Метою розроблення АСКП є поліпшення системи керування підприємством і, як наслідок, одержання більш високої якості продукції, що випускається, з меншими витратами.

АСКТП призначені для розв'язування задач керування технологічним процесом з *обов'язковою* участю людини - *оператора*.

Ці системи використовують у тому випадку, коли за будь-якими причинами неможливо автоматизувати усі задачі керування, і тоді для їх розв'язування звертаються до людини. Вона звичайно приймає остаточне рішення, а завчасне оброблення інформації та її збирання здійснюють, як правило, цифрові пристрої та машини.

В АСКТП більшість контурів регулювання будують за ієрархічним принципом.

Перший нижній рівень ієрархії — це основні регулятори, які стабілізують технологічні параметри або змінюють їх відповідно до керуючих сигналів. Основні регулятори, як правило, безпосередньо впливають на виконавчі органи.

Другий рівень утворюють коректуючі регулятори, які керують основними регуляторами, тим самим непрямо впливаючи на технологічний процес.

На більш високих рівнях регулювання в АСКТП знаходяться обчислювальні комплекси, які прораховують

оптимальні режими і змінюють завдання регуляторам, що знаходяться на нижніх рівнях ієрархії.

У випадку повної або майже повної автоматизації використовуються системи *автоматичного* керування (САК).

Комплекс пристроїв, призначених для автоматичної підтримки бажаного режиму роботи об'єкта керування, називається системою *автоматичного* керування.

Метою автоматичного керування, як правило, є підтримка заданих значень керованих (регульованих) величин при повній автоматизації. Ця мета досягається за допомогою об'єднаних у систему автоматичних пристроїв, працюючих без втручання людини. Вони розв'язують більш прості завдання ніж автоматизовані системи, однак з більшою швидкістю і точністю. Ці системи звичайно входять, як складова частина, в автоматизовані системи керування, звільняючи людину від рутинної роботи і дають їй час для прийняття відповідальних рішень.

Так, наприклад, у сучасних автомобілях багато операцій, що виконувались раніше людиною, передаються автоматом. Це, наприклад, підтримка заданої температури та вологості повітря в салоні автомобіля, автоматичне перемикання швидкості, вибір і підтримка оптимальної швидкості руху автомобіля.

## Лекція 5. Загальна характеристика керуючих автоматів

Керуючі автомати призначені для керування об'єктами виробничого призначення і технологічними процесами в автоматичному режимі роботи. Вони входять як складова і важлива частина в керуючі системи і забезпечують виконання їх алгоритмів керування. Крім керуючих автоматів, в керуючу систему входять датчики, електроавтоматика, засоби зв'язку, інформаційна система, виконавчі та регулюючі органи. Поряд із основною функцією керування керуючі автомати вирішують також завдання забезпечення такого режиму системи керування, в якому кожна команду можна здійснити за допомогою органів ручного керування. Керуючі автомати також здійснюють типові захисти, такі, як захист від самозапуску при зникненні та появи знову живлення від електромережі, захист від перенавантажень, захист при несправностях або неправильних діях оператора, контроль справності пристрою, локалізацію та усунення несправностей, сигналізацію про хід роботи обладнання і в особливих випадках видачу сигналів у систему реєстрації та обліку роботи обладнання.

Керуючі автомати, крім шафи керування, звичайно, містять одну або декілька шаф електроавтоматики, які з'єднуються з обладнанням, що обслуговується, кабелями або джгутами проводів. Шафи керування і електроавтоматики виготовляють і монтують, як правило, на спеціальних підприємствах. Після підключення їх до об'єктів керування вони повинні бути повністю готові до роботи.

Для розроблення керуючих автоматів повинні бути відомі кількість і тип таких виконавчих елементів, як електродвигуни, електромагніти, муфти, нагрівачі, а також інформаційних елементів: шляхових і кінцевих перемикачів, різних датчиків і одночасно з цим необхідно знати алгоритми керування, які уточнюються на всіх етапах розроблення і дуже часто змінюються на етапі налагодження системи, а інколи навіть і в процесі експлуатації.

Керуючі автомати можуть бути електричними, пневматич-

ними, гідравлічними, електро- і пневмогідравлічними.

Крім того, вони підрозділяються на системи із *логічним* (релейним) і *аналоговим* принципами керування. Останні у своїй роботі поряд із цифровою обробкою використовують й аналогове оброблення інформації. Вони застосовуються у системах керування на підприємствах із неперервним циклом виробництва і є, як правило, більш складними, ніж перші.

Логічні керуючі автомати, які часто ще називаються *дискретними*, або *цифровими*, використовуються для керування виробничими об'єктами з дискретним режимом роботи типу ввімкнено-вимкнено.

Початково у логічних керуючих автоматах використовувались релейно-контактні схеми зі схемною (жорсткою) логікою. Наступним етапом у розвитку керуючих автоматів було використання в них безконтактних логічних елементів від магнітних і напівпровідникових до інтегральних спочатку малого ступеня інтеграції, потім середнього, великого і надвеликого.

У результаті виникла можливість реалізації в одній мікросхемі повної схеми того чи іншого керуючого автомата. Такі мікросхеми називались *замовленими*. Однак при цьому виникло протиріччя між економічною доцільністю випуску таких схем великими серіями і, як правило, невеликою їх кількістю при вирішенні того чи іншого конкретного практичного завдання керування.

Ці суперечності вдалося вирішити розробленням спеціальних великих інтегральних схем, названих *мікропроцесорами*. Їх можна було націлювати на вирішення практичних завдань програмно.

У результаті були розроблені *програмовані* керуючі автомати. У порівнянні з раніше існуючими схемними рішеннями виникли додаткові можливості їх модернізації, додання нових функцій, що часто потрібно в процесі експлуатації керованих об'єктів. Однак одержання цих переваг спричинило зниження швидкодії та частково надійності. Вартість програмованих (гнучких) керуючих автоматів у ряді випадків також була

більшою за вартість пристроїв із жорсткою логікою на інтегральних схемах з малим ступенем інтеграції. Тому хоча програмовані керуючі автомати й потіснили значно автомати із жорсткою логікою, однак у кінцевому підсумку витіснити їх не змогли. Більше того, в даний час із появою нових інтегральних технологій вартість і час виготовлення замовлених мікросхем настільки знизилась, що у ряді випадків їх використання є економічно більш вигідним для завдань керування, ніж універсальних. Тому часто при виконанні відповідальних замовлень йде зворотний процес - заміна у ряді випадків програмованих керуючих автоматів автоматами на замовлених великих інтегральних схемах.

Таким чином, сучасні керуючі автомати поділяються на два основних класи - із *програмованою (гнучкою)* і зі *схемною (жорсткою)* реалізацією алгоритмів керування.

У схемно-реалізованих автоматах команди керування відпрацьовуються за допомогою спеціальних схем. Такі пристрої не мають достатньо упорядкованої структури. Їх складність визначається кількістю команд керування і зв'язками між ними. Чим більше цих команд, тим складніша реалізація автомата.

У загальному випадку автомати такого типу надто складні у проектуванні і насамперед тому, що вони не мають регулярної структури. Її відсутність призводить у випадку змін в алгоритмі керування до необхідності вагомого перероблення автомата. На практиці буває простіше побудувати новий пристрій для нової послідовності команд керування, ніж вносити зміни у раніше розроблений автомат.

Крім того, автомати зі схемною реалізацією незручні в експлуатації, тому що кожен з них має у своїй структурі особливості, які необхідно знати при їх експлуатації.

Зазначені недоліки обмежують можливості використання автоматів зі схемною логікою для складних завдань керування.

У той самий час для вирішення більш простих завдань керування вони широко застосовуються через низьку вартість, високу швидкодію та надійність.

Однак існує ще один важливий клас керуючих автоматів,

що займає проміжне місце між програмованими автоматами і автоматами із схемною логікою, які використовують мікропрограмний принцип керування - мікропрограмні керуючі автомати. Цей принцип вперше був використаний при побудові ЕОМ для підвищення регулярності їх структури.

В основу мікропрограмних керуючих автоматів покладена ідея мікропрограмного керування на рівні мікрооперацій.

Мікропрограмний автомат є програмованим автоматом, в якому на відміну від звичайної або керуючої ЕОМ програмування ведеться на самому низькому рівні - машинній мові.

Програмування мікропрограмного пристрою аналогічне до звичайного програмування з тією різницею, що програми пишуться машинною мовою.

Укладач мікропрограм повинен чітко розуміти функції кожного блоку універсального мікропрограмного пристрою. Він повинен знати усі вхідні і вихідні сигнали для кожної частини автомата, знати часові характеристики сигналів і схеми, на які він впливає, тому що мікропрограма керує пристроєм на рівні схемної реалізації. Це означає, що розробник мікропрограмного пристрою керування повинен бути одночасно і програмістом, і схемотехніком.

Таким чином, мікропрограмування це є деяка методика з реалізації функцій керування, в основу якої покладене упорядковане розроблення керуючих сигналів пристроєм керування за раніше заданою програмою.

Основні переваги мікропрограмного керування - це:

1. Стандартна структура пристрою керування, що вирішує різні завдання керування.

2. Гнучкість пристроїв керування, яка реалізується їх перепрограмуванням, що зберігає час при розробленні пристроїв керування.

3. Простота процедури діагностики, виявлення та усунення несправностей.

4. Простота проектування, налагодження та експлуатації.

5. Великий термін служби, що забезпечується за рахунок перепрограмування мікропрограмних пристроїв керування в

в процесі їх експлуатації.

У той самий час мікропрограмні автомати потребують більше часу для своєї роботи, ніж автомати зі схемною логікою, тому що потрібен додатковий час для послідовного виконання операцій, як це здійснюється в ЕОМ.

Мікропрограмні керуючі автомати поєднують у собі переваги універсальних програмованих керуючих автоматів і автоматів зі схемною логікою. Вони, з одного боку, мають необхідну гнучкість та універсальність, а з іншого - потребують для своєї реалізації відносно невеликої кількості апаратних витрат і відрізняються високою швидкістю і надійністю.

Тому для нескладних завдань керування їх використання може бути цілком ефективним.

Треба також зазначити, що віртуальні схеми мікропрограмних автоматів та автоматів зі схемною логікою використовуються при програмуванні мовами високого рівня, які широко застосовуються у сучасних програмованих *мікропроцесорних* системах керування.

Такі системи знайшли широке розповсюдження для розв'язання складних виробничих завдань керування. Вони є спеціалізованими керуючими ЕОМ, об'єднаними між собою каналами надійного зв'язку і програмовані мовами високого рівня за допомогою універсальних ЕОМ. Після програмування програма з універсальної ЕОМ переписується в керуючу і таким чином остання вирішує конкретне завдання керування. Далі керуюча машина вмикається в об'єкт керування і працює за заданою програмою. За необхідності керуючу програму модернізують за допомогою спеціальних пристосувань.

Таким чином, сучасні керуючі автомати можна поділити на автомати зі схемною і гнучкою логікою, Автомати зі схемною логікою призначені для вирішення простих завдань керування, а з гнучкою - для більш складних. Ті та інші автомати можуть бути як аналоговими, так і цифровими. Останні на сьогодні є найбільш поширеними і в поєднанні з мікропроцесорною технікою найбільш ефективними для вирішення складних завдань керування.



## Розділ II. Подання алгоритмів роботи керуючих автоматів

### Лекція 6. Подання роботи керуючих автоматів граф-схемами алгоритмів

При описі функціонування різних засобів обчислювальної техніки досить часто використовується їх подання у вигляді сукупності керуючого і операційного автоматів (див. рис. 2.1).

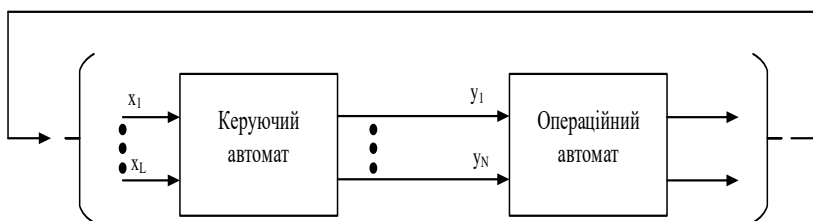


Рисунок 2.1 – Структура керуючого автомата

Так, в ЕОМ до операційного автомату слід віднести блоки пам'яті, регістри, суматори, канали передачі інформації, шифратори і таке інше, тобто всі пристрої, які виконують деякі операції, а до керуючого автомату - ту частину ЕОМ, яка координує дії перерахованих пристроїв, визначаючи послідовність обробки в них інформації.

**Завданням** керуючого автомата є, таким чином, вироблення розподіленої в часі послідовності вихідних (керуючих) сигналів, під впливом яких в операційному автоматі здійснюється деяка операція. Послідовність таких операцій (дій) називається алгоритмом. Алгоритм зручно подавати за допомогою граф-схем алгоритмів (ГСА).

**Означення:** Графічне подання роботи автомата, у якому кожний з його етапів зображений геометричною фігурою, які з'єднані стрілками, що показують послідовність виконання

етапів, за необхідності з поясненнями, називається *граф-схемою алгоритму (ГСА)*.

Інколи дають ще одне *означення*: ГСА називають зв'язний орієнтовний граф, що містить одну початкову вершину (Початок), одну кінцеву вершину (Кінець) та довільну множину умовних та операторних вершин.

У початковий оператор, який інколи позначають  $A_0$  не входить жодна дуга, а виходить із нього тільки одна. У кінцевий оператор  $A_k$  можуть входити одна або декілька дуг, але не виходить жодна дуга.

Оператори, які розміщені між операторами початку і кінця, можна поділити на два типи: керуючі оператори – оператори, за якими формуються вихідні керуючі сигнали, і логічні – оператори, за якими перевіряються логічні умови.

Вершини керуючих операторів відображають за допомогою прямокутників, в середині яких прописують словами керуючу команду або позначають символічно великими літерами латинського алфавіту, наприклад,  $A_1, A_2, \dots, A_j$ .

Логічні оператори називають ще умовними. Їх вершини відображають ромбами, що містять умову, яка записана словами або символічно позначена маленькими літерами латинського алфавіту:  $x_i, p_i, i = 1, 2, \dots, n$ .

Керуюча вершина може мати тільки один вихід, в той же час логічна вершина має два виходи, які позначають, як правило, 0 і 1. Інколи їх заміняють словами «Ні» і «Так» відповідно. Це впливає з того, що результатом перевірки деякої логічної умови  $x$  можливі два виходи:  $x = 1$  або  $x = 0$ .

Зображення вершин ГСА наведені на рисунку 2.2.

***ГСА повинна задовольняти наступним умовам:***

1. Має одну початкову і одну кінцеву вершину, які не нумеруються.
2. Містить кінцеву кількість вершин, кожна з яких є або початковою, або кінцевою, або керуючою, або логічною.

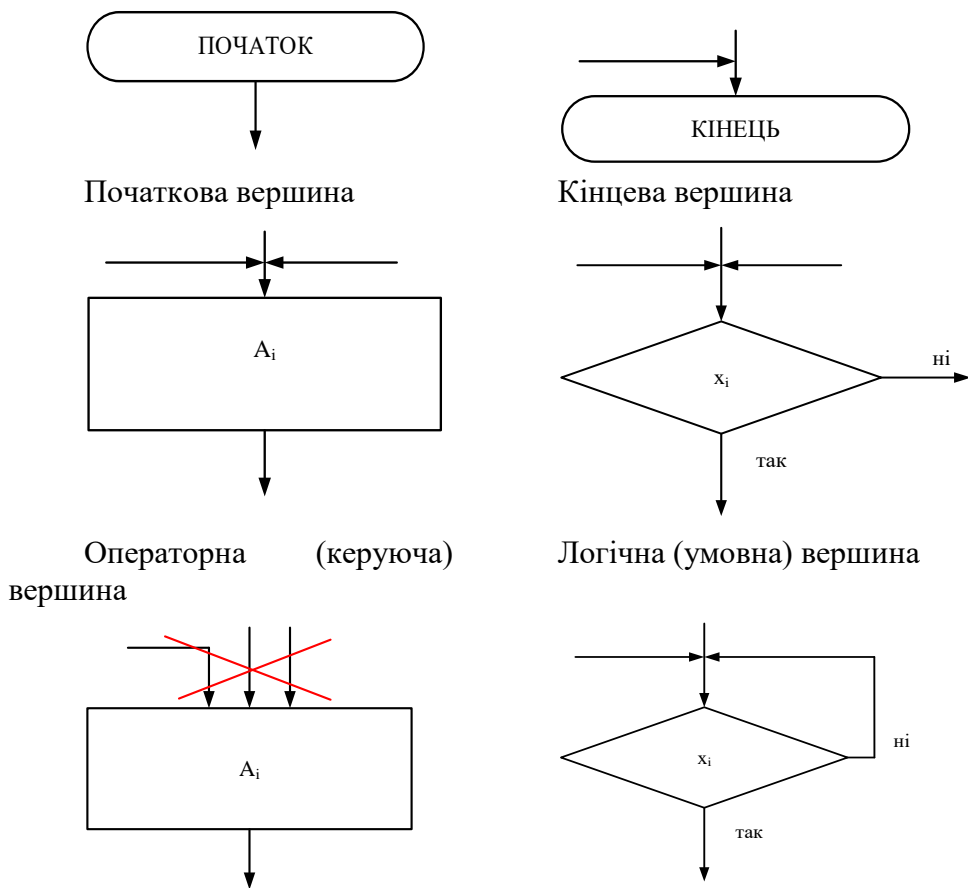


Рисунок 2.2 – Зображення вершин ГСА

3. Входи і виходи вершин з'єднуються між собою за допомогою дуг, направлених завжди від виходу до входу.
4. Кожний вихід об'єднаний тільки з одним входом.
5. Будь-який вхід об'єднується не менше ніж з одним виходом.
6. Будь-яка вершина ГСА лежить, принаймні, на одному шляху з вершини «Початок» у вершину «Кінець».
7. Один з виходів умовної вершини може бути з'єднаним з її входом, що неприпустимо для керуючої вершини.

8. В кожній умовній вершині записується логічна умова з множини логічних умов. Дозволяється в різних умовних вершинах записувати однакові логічні умови.

9. В кожній операторній вершині записується оператор, що являє собою вихідний сигнал або сукупність вихідних сигналів керуючого автомата.

Запис у вершині може бути символічним або змістовним.

Зазвичай при проектуванні пристроїв попередньо складають змістовну ГСА, в якій в середині умовних та керуючих вершин записані логічні умови та мікрооперації в змістовних термінах.

На рисунку 2.3 наведено приклад фрагменту змістовної ГСА.

Змістовну граф-схему алгоритму одержують на основі аналізу мовного опису умов роботи керуючого автомата.

Спочатку зображується початкова вершина ГСА. Після неї одержують першу операторну вершину, яка є, як правило, оператором введення початкових даних. Потім викреслюються наступні вершини, які є як керуючими, так і логічними операторами, і усі зв'язки між ними. Останньою відображається кінцева вершина.

У вершинах проставляються змістовні означення команд, як це показано на рис. 2.3. У результаті одержують змістовну ГСА.

За необхідністю здійснюється перехід до символічної ГСА шляхом заміни змістовних операторів символами. Така ГСА буде мати більш простий вигляд, ніж змістовна, в чому і є її основна перевага.

На рисунку 2.4 наведена символічна ГСА для змістовної ГСА з рисунку 2.3.

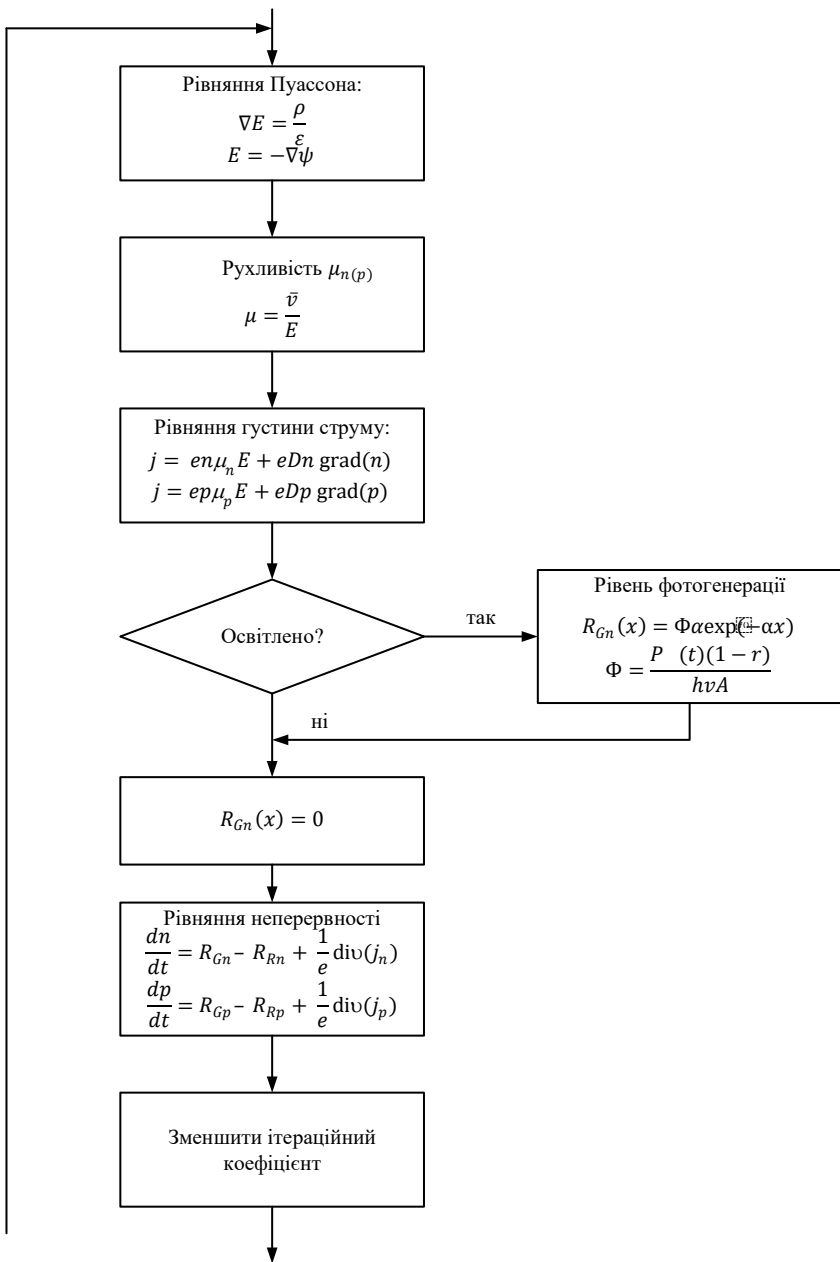


Рисунок 2.3 – Фрагмент змістовної ГСА

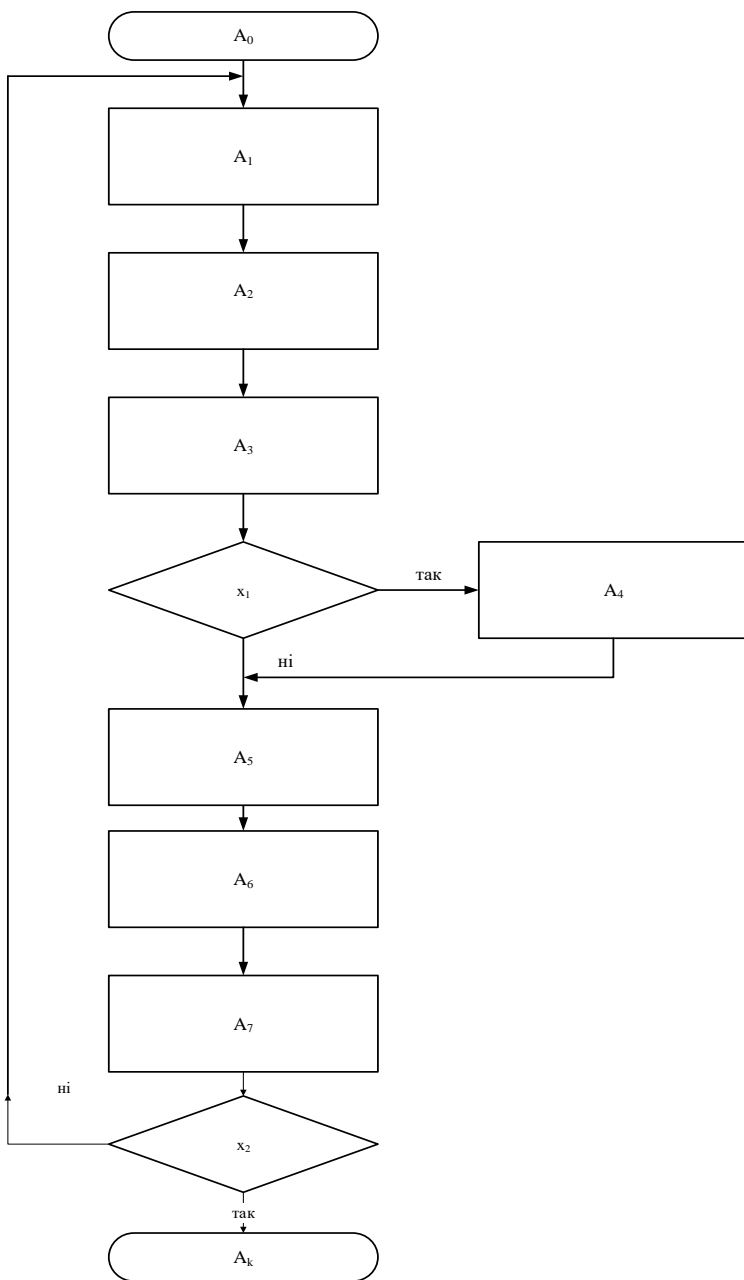


Рисунок 2.4 – Символьна ГСА

## Класифікація ГСА

**Означення.** ГСА, в якій відсутні логічні оператори, називається *простою* або *лінійною*.

На рис. 2.5 наведений приклад лінійної ГСА для обчислення функції, яка не має умов.



Рисунок 2.5 – Лінійна ГСА

**Означення.** ГСА, в якій є логічні вершини, називається *розгалуженою*.

На рисунку 2.6 наведено приклад розгалуженої ГСА для обчислення функції, яка має наступні умови:

$$\begin{cases} y = bx^2 + d & \text{при } b > 0, \\ y = bx^2 + cx + d & \text{при } b \leq 0. \end{cases}$$

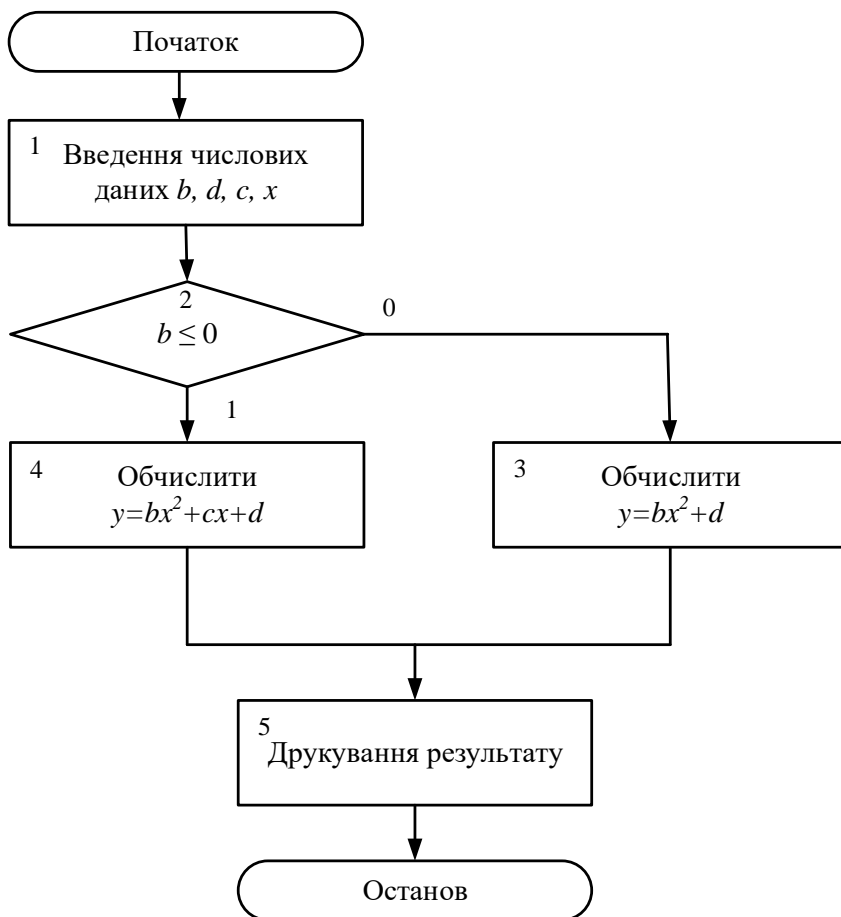


Рисунок 2.6 – Розгалужена ГСА

**Означення.** ГСА, в якій організуються багаторазові обчислення за однією і тією самою схемою з даними, що змінюються, називається *циклічною*.

На рисунку 2.7 наведено приклад циклічної ГСА для функції:

$$y = \sqrt{\sin x} + cx,$$

в якій обчислення починаються з  $x_{min}$  із кроком  $\Delta x$  до  $x_{max}$ .





Рисунок 2.7 - Циклічна ГСА для обрахунку функції з даними, що змінюються

**Означення 5.** ГСА, яка містить розгалуження, цикли і операторні вершини, називається *комбінованою* або *універсальною*. Приклад такої ГСА наведено на рисунку 2.8.

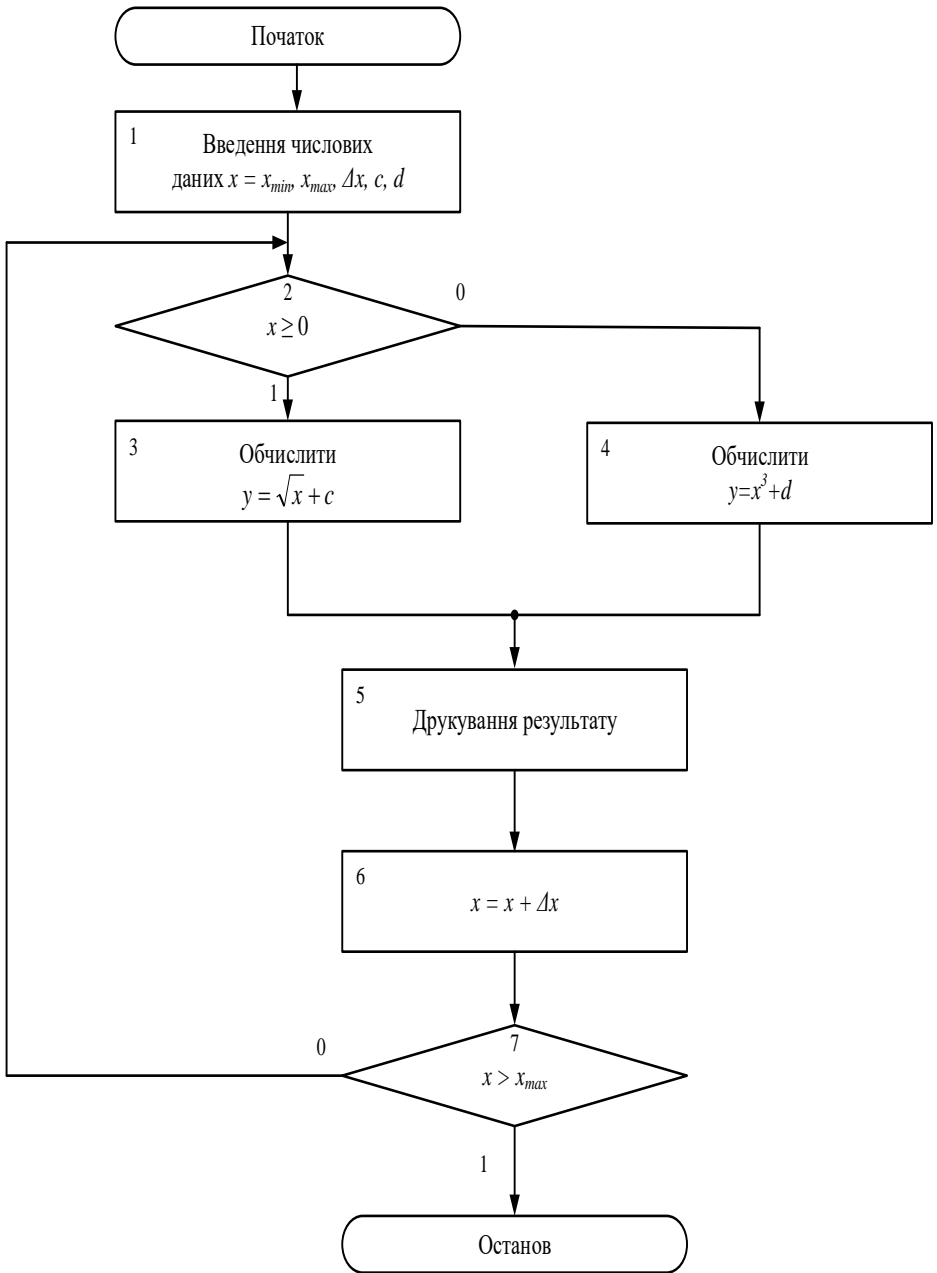


Рисунок 2.8 – Комбінована ГСА

Комбінована граф-схема алгоритму з рисунку 2.8 описує процедуру обрахунку функції

$$\begin{cases} y = \sqrt{x} + c & \text{при } x \geq 0, \\ y = x^3 + d & \text{при } x < 0, \end{cases}$$

в якій обчислення починаються з  $x_{min}$  із кроком  $\Delta x$  до  $x_{max}$ .

### ***Завдання з зірочкою)))***

- Побудувати змістовну та символічну ГСА керуючого автомату з розміну монет. При цьому будемо вважати, що можна розмінювати монети номіналом в 10, 15, 20, 25 і 50 копійок, і кожна монета розмінюється на відповідну кількість монет номіналом в 5 копійок кожна. Будемо вважати, що в якості операційних автоматів використовуються:

- блок аналізу вартості монет, що розмінюються;
- блок видачі монет;
- блок аналізу наявності монет;
- лічильник монет.

Для заданої умови функціонування можна побудувати не єдину ГСА, тому запропоновані Вами варіанти повинні відрізнятися між собою.

- Запропонувати свій варіант пристрою, побудувати для нього змістовну та символічну ГСА, структурну схему пристрою та навести докладний опис алгоритму функціонування.

## Лекція 7. Подання роботи керуючих автоматів логічними схемами алгоритмів

Граф схема алгоритму (ГСА) відрізняється наочністю та зручністю подання алгоритму функціонування, однак займає для свого зображення багато місця і зовсім непридатна для введення в ЕОМ. Тому часто після подання керуючого алгоритму у вигляді ГСА потрібно його перетворення в ЛСА.

**Означення.** Вираз, складений із керуючих та логічних операторів, які записані один за одним, а також певним чином розставлених пронумерованих стрілок, називається *логічною схемою алгоритму (ЛСА)*.

ЛСА визначає порядок виконання операторів (керуючих і логічних) залежно від логічних умов, що до неї входять.

Якщо порядок виконання керуючих операторів в ЛСА строго фіксований, то описуваний алгоритм буде *лінійним*.

Якщо порядок виконання керуючих операторів залежить від умов, то такий алгоритм буде *розгалуженим*.

Керуючі оператори визначаються великими літерами латинського алфавіту, наприклад,  $A, B, C, \dots$  або однією й тією самою великою літерою, але з різними індексами, наприклад,  $A_1, A_2, A_3, \dots$  або  $A_i, i = 1, 2, 3$ . Кількість індексів може дорівнювати двом, наприклад,  $i, j$ . Тоді оператор має вигляд  $A_{ij}$ .

Логічні оператори визначаються малими літерами  $x_1, x_2, \dots, x_n$  і можуть набувати тільки двох значень 1 або 0.

Оскільки кожна логічна умова може набувати тільки одного із двох значень 1 або 0, то потрібен символ, який би зазначав подальший порядок виконання операцій. Тому після кожної логічної умови стоїть стрілка, яка має угорі номер. Ця стрілка має початок  $\uparrow^i$  і кінець  $\downarrow^i$ . Початок стрілки стоїть безпосередньо справа біля логічної умови, що позначається літерою  $x_i$ , або  $p_i$  з індексом унизу. Її номер, що стоїть безпосередньо над стрілкою справа та її напрямок показує, куди повинно бути передане керування алгоритмом у випадку, якщо логічна умова  $x_i = 0$ . Кінець стрілки, що має також над собою справа номер, показує оператор, якому повинно бути у цьому випадку передане

керування. Цей оператор стоїть відразу після кінця стрілки справа. Якщо  $x_i = 1$ , то після логічної умови виконується оператор, який стоїть поряд справа. Це може бути або керуючий оператор, або логічна умова.

***ЛСА повинна задовольняти таким умовам:***

1. Містить один початковий і один кінцевий оператор, які не мають перед собою і після себе стрілок.
2. Після кожної логічної умови стоїть початок стрілки  $\uparrow^i$ .
3. Не може бути двох і більше початків і кінців стрілок з однаковими індексами.
4. Для кожного кінця стрілки  $\downarrow^i$  повинен бути хоча б один початок стрілки  $\uparrow^i$ .
5. Для кожного початку стрілки  $\uparrow^i$  повинен бути точно один кінець стрілки  $\downarrow^i$ .

Серед логічних операторів може бути оператор  $\omega$ , який не потребує перевірки логічної умови. Це безумовний оператор. Він передає виконання операції оператору, номер якого і зазначений над початком стрілки  $\uparrow^i$ .

Робота алгоритму починається з того, що виконується самий лівий оператор ЛСА. Після нього визначається подальший справа оператор. Якщо це керуючий оператор, то здійснюється його виконання. Якщо це логічний оператор, то можливі два випадки:

1. Логічна умова має одиничне значення. У цьому випадку виконується оператор, що стоїть справа поряд.
2. Логічна умова дорівнює нулю. Виконується оператор, на який показує стрілка, що починається після даної умови.

Робота ЛСА закінчується, коли останній із операторів, що виконується, містить вказівку про зупинення роботи алгоритму або коли на деякому етап не виявляється оператор ЛСА, який повинен був би виконуватись.

Таким чином, порядок виконання операторів в ЛСА визначається послідовністю їх розміщення у виразі, а також розподілом нумерованих стрілок логічних умов, що входять до неї.

## Лекція 8. Взаємні перетворення між граф-схемами і логічними схемами алгоритмів

### 1. Побудова ЛСА за заданою ГСА

Як уже зазначалося вище, граф-схема алгоритму (ГСА) відрізняється наочністю та зручністю подання, однак займає для свого зображення надто багато місця і непридатна для введення в ЕОМ, У той самий час логічна схема алгоритму (ЛСА) подається компактно у вигляді рядка, який складається з операторів, що послідовно виконуються, і тому зручна для кодування і запису в ЕОМ. Тому часто після подання керуючого алгоритму у вигляді ГСА потрібно його перетворення в ЛСА.

*Алгоритм перетворення ГСА в ЛСА містить наступні кроки:*

- Аналізується ГСА, для якої необхідно записати ЛСА. Вибирається шлях максимальної довжини.
- Першим лівим оператором в рядку ЛСА записується початковий оператор.
- Праворуч від написаного оператора записують наступний в ГСА за ним керуючий оператор або логічна умова.
- Якщо в ГСА оператори, що виконуються, слідує один за одним, то відповідні оператори аналогічно один за одним записуються в ЛСА.
- Якщо наступним оператором ГСА, що аналізується, є логічна умова, то, записавши його в ЛСА, праворуч від нього обов'язково проставляють стрілку вгору (початок стрілки) справа вгорі якої проставляється номер. У загальному випадку номер стрілки не збігається з індексом логічної умови. Праворуч від стрілки записується оператор, управління до якого передається в разі рівності одиниці логічної умови. Передача управління в разі рівності логічної умови нулю здійснюється за стрілкою. Кінець стрілки з відповідним номером проставляється перед оператором, якому передається керування.

- Описана процедура повторюється до тих пір, поки в рядку ЛСА не буде записаний кінцевий оператор  $A_k$ .

- При аналізі більш складних ГСА іноді виникають ситуації, коли жоден з можливих шляхів не проходить через всі вершини ГСА. Формування вставок в ЛСА для операторних вершин, що випадають при аналізі, здійснюється за допомогою символу безумовного переходу  $\omega$ , який виконує роль знаків пунктуації в ЛСА.

Для перевірки правильності побудови ЛСА треба розв'язати зворотню задачу переходу від ЛСА до ГСА.

**Приклад.** За заданою на рисунку 2.9 ГСА побудувати ЛСА керуючого автомата.

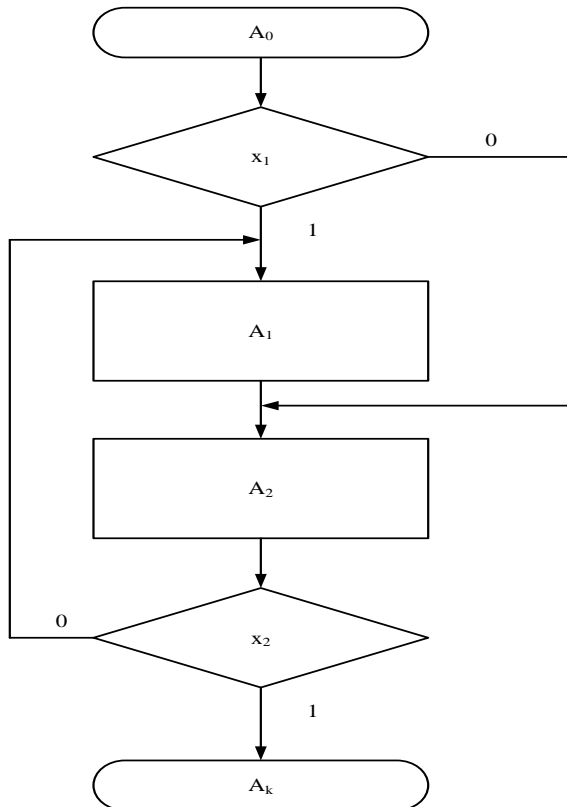


Рисунок 2.9 – Граф-схема алгоритму

**Розв'язування.** Із аналізу ГСА виходить, що її початковою вершиною є  $A_0$ . Отже, першим оператором ЛСА буде оператор  $A_0$ . Від вершини  $A_0$  на ГСА іде зв'язок до логічної вершини (умовного оператора)  $x_1$ . Тому після оператора  $A_0$  в ЛСА повинен стояти оператор  $x_1$ . Після умовного оператора обов'язково ставимо початок стрілки  $\uparrow^1$ . Треба відмітити, що номер стрілки не має особливого значення і вибирається довільно. Єдине, що потрібно, щоб не було двох стрілок з однаковими номерами. Справа після стрілки проставляється оператор, якому передається керування якщо логічна умова, яка перевіряється, виконується, тобто дорівнює одиниці:

$$A_0 x_1 \uparrow^1 A_1$$

Після оператора  $A_1$  в ГСА виконується оператор  $A_2$ , тому і в ЛСА після оператора  $A_1$  запишемо оператор  $A_2$ :

$$A_0 x_1 \uparrow^1 A_1 A_2$$

Після оператора  $A_2$  знову перевіряється логічна умова, це умовний оператор  $x_2$ . Після умовного оператора необхідно поставити початок стрілки зі своїм номером:

$$A_0 x_1 \uparrow^1 A_1 A_2 x_2 \uparrow^2$$

У випадку виконання умови  $x_2$ , тобто коли  $x_2 = 1$  керування передається кінцевому оператору  $A_k$ . Цей оператор є останнім в ГСА, аналогічно він повинен бути останнім і в ЛСА:

$$A_0 x_1 \uparrow^1 A_1 A_2 x_2 \uparrow^2 A_k$$

Тепер записані всі операторні вершини. Залишилось проставити кінці стрілок, тобто зазначити яким операторам буде передано управління, якщо логічна умова, що перевіряється не буде виконана. Якщо  $x_1 = 0$ , тобто умова  $x_1$  не виконалась, то керування передається оператору  $A_2$ , і перед цим оператором необхідно поставити кінець стрілки з номером 1.



$$A_0 x_1 \uparrow^1 A_1 \downarrow^1 A_2 x_2 \uparrow^2 A_k$$

Якщо  $x_2 = 0$ , тобто умова  $x_2$  також не виконалась, то керування передається оператору  $A_1$ , і перед цим оператором необхідно поставити кінець стрілки з номером 2:

$$A_0 x_1 \uparrow^1 \downarrow^2 A_1 \downarrow^1 A_2 x_2 \uparrow^2 A_k$$

На наступному кроці перевіряють пари стрілок – не повинно бути двох стрілок з однаковими номерами, а також кожен початок стрілки повинен мати кінець стрілки.

Отримана ЛСА є остаточною.

### **Завдання з зірочкою)))**

За заданими на рисунках 2.10 – 2.14 ГСА побудувати ЛСА керуючого автомата.

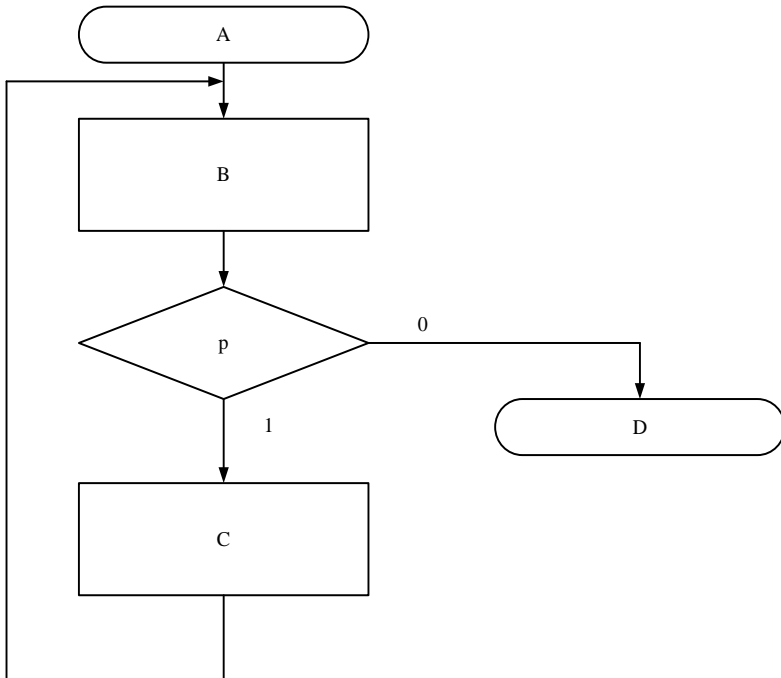


Рисунок 2.10 – Граф-схема алгоритму

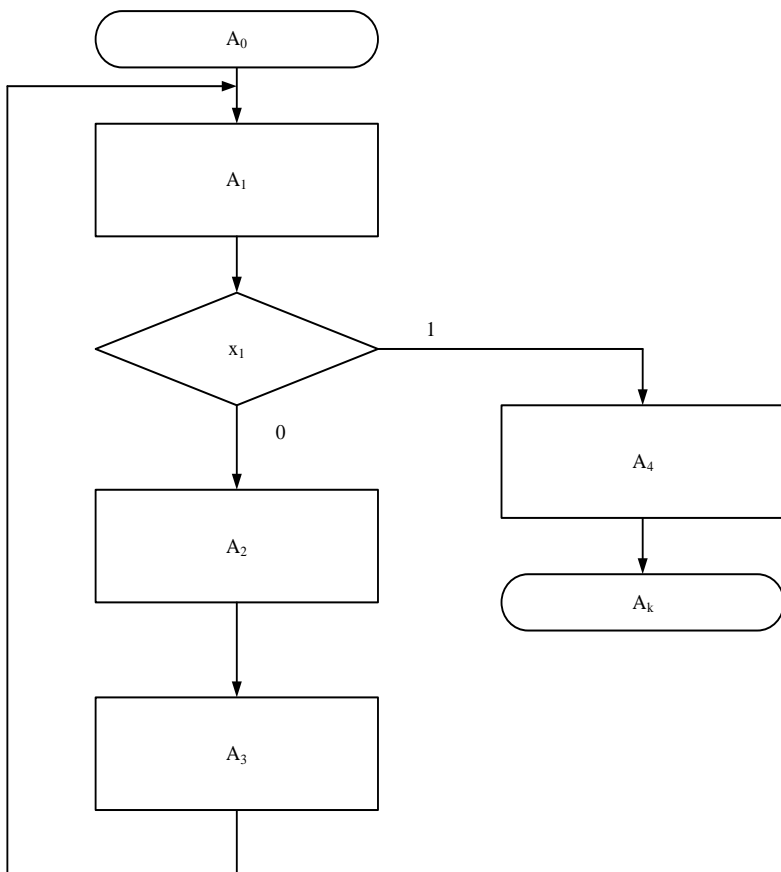


Рисунок 2.11 – Граф-схема алгоритму

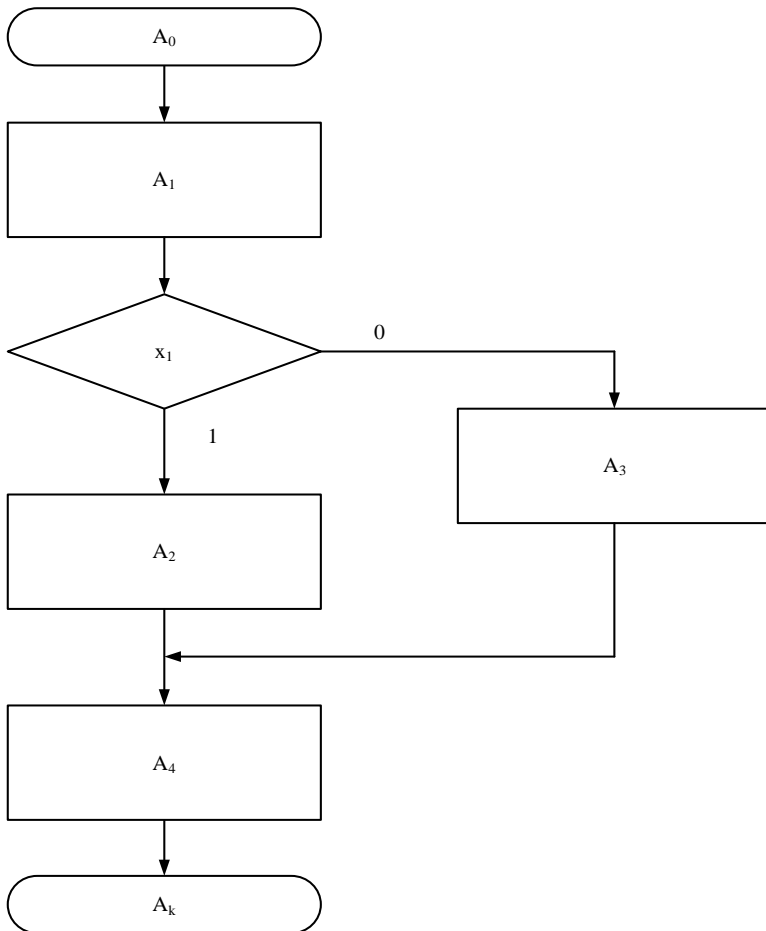


Рисунок 2.12 – Граф-схема алгоритму

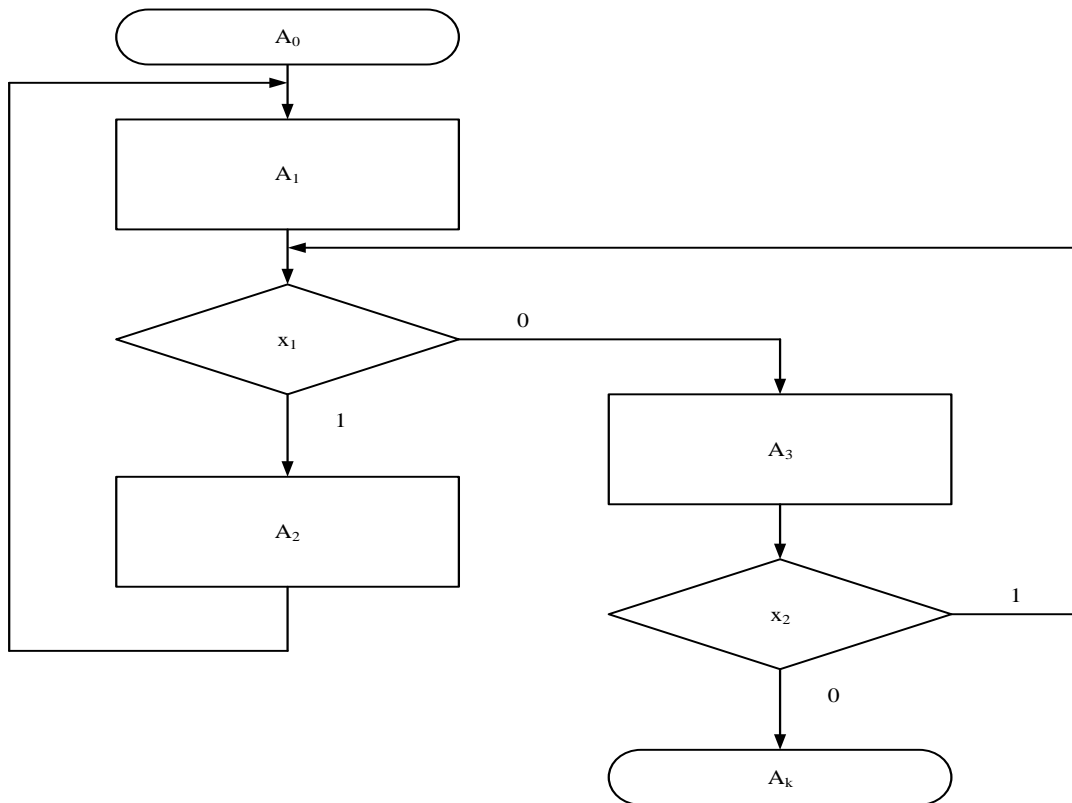


Рисунок 2.13 – Граф-схема алгоритму

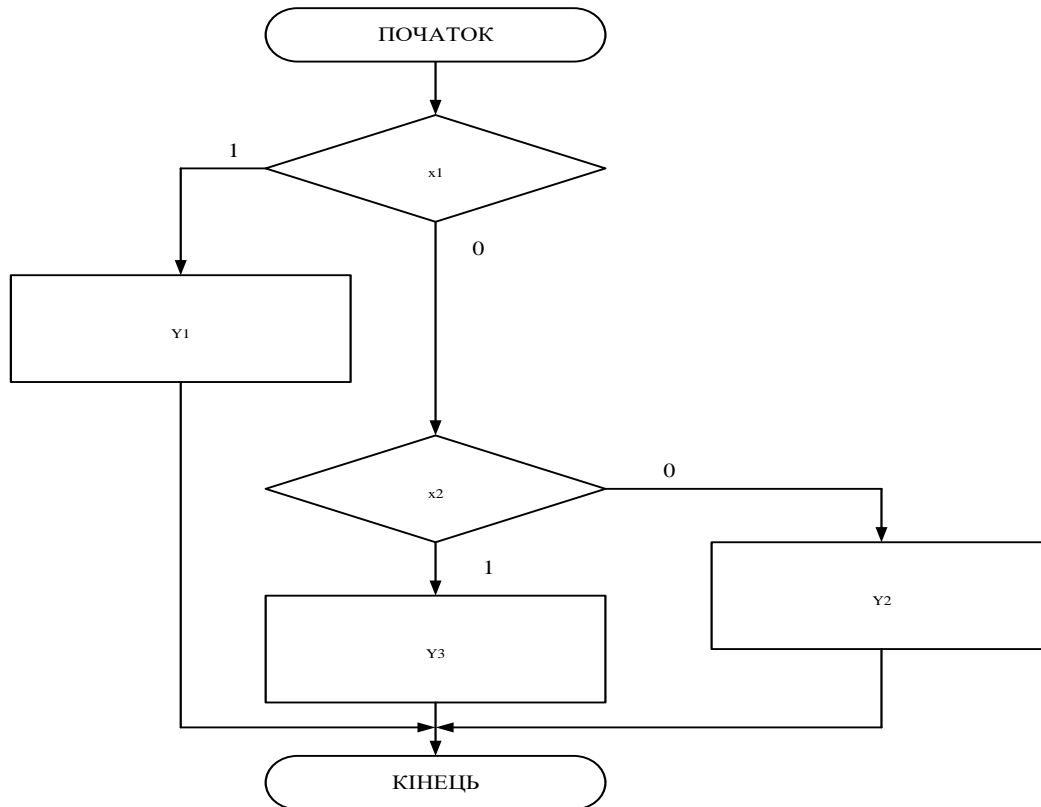


Рисунок 2.14 – Граф-схема алгоритму

## 2. Побудова ЛСА за заданою ГСА

Недоліком ЛСА є недостатня її наочність. Тому часто для перевірки правильності ЛСА переходять від неї до ГСА, яка є найбільш наочною з усіх можливих форм запису алгоритмів.

Скористаємося алгоритмом переходу від ЛСА до ГСА, при цьому повинні виконуватися наступні умови:

- Відповідна ГСА має одну початкову і одну кінцеву вершини, а число операторних вершин дорівнює числу операторів в ЛСА.

- З ЛСА визначається оператор «Початок» і викреслюється відповідна йому початкова вершина ГСА.

- Аналізується наступний за початковим оператор в ЛСА. Це може бути як керуючий оператор  $A_i$ , так і логічна умова  $x_i$ .

- Вихід початкової вершини ГСА з'єднується дугою з входом вершини, що відповідає в ЛСА найближчому розташованому праворуч від  $A_i$  оператору або логічній умові. Аналогічним чином в ГСА проводяться дуги з інших операторних вершин.

- Одиначний вихід умовної вершини  $x_i$  ГСА з'єднується з входом вершини, розташованої в ЛСА праворуч від  $x_i$ . Після логічної умови  $x_i$  в ЛСА завжди стоїть верхня стрілка, і, відповідно, нульовий вихід умовної вершини  $x_i$  ГСА з'єднується з входом вершини (оператора або логічної умови), перед якою стоїть кінець стрілки з аналогічним номером.

- Аналогічно будуються наступні оператори ЛСА, з'єднуються дужками, поки не будуть подані в ГСА усі оператори ЛСА.

- Щоб переконатися у правильності побудови ГСА за заданою ЛСА, необхідно розв'язати зворотну задачу – за побудованою ГСА одержати ЛСА.

**Приклад.** За наступною ЛСА побудувати ГСА керуючого автомата:

$$A_0 A_1 \downarrow^2 x_1 \uparrow^1 A_2 \downarrow^1 A_3 x_2 \uparrow^2 A_4 A_k$$

**Розв'язування.** Очевидно, що оператором «Початок» є оператор  $A_0$ . Будуємо відповідну вершину на ГСА. За оператором  $A_0$  справа в ЛСА записаний оператор  $A_1$ , це керуючий оператор, тому в ГСА побудуємо відповідний прямокутник і дугу, що виходить з оператора  $A_0$  на вхід оператора  $A_1$ . Далі із ЛСА виходить, що після виконання оператора  $A_1$  іде слідом логічний оператор  $x_1$ , який у випадку  $x_1 = 1$  передає керування оператору  $A_2$  і у випадку  $x_1 = 0$  - оператору  $A_3$ . Будуємо вершини операторів  $x_1, A_2, A_3$  на ГСА і пов'язуємо відповідні виходи вершини  $x_1$  із входами вершин  $A_2$  і  $A_3$ . Потім пов'язуємо вихід вершини  $A_2$  із входом вершини  $A_3$ . Після оператора  $A_3$  в ЛСА слідує логічна умова  $x_2$ , будуємо відповідний ромб, який повинен мати два виходи. У випадку  $x_2 = 1$  умовний оператор  $x_2$  передає керування записаному після нього справа в ЛСА керуючому оператору  $A_4$ , відповідно креслимо дугу, яка пов'яже одиничний вихід логічного оператора  $x_2$  із входом керуючого оператора  $A_4$ . У випадку  $x_2 = 0$  умовний оператор  $x_2$  передає керування оператору  $x_1$ . Це витікає з того, що початок стрілки з номером два пов'язаний з кінцем стрілки з таким самим номером, який стоїть перед оператором  $x_1$ . Будуємо відповідну дугу. Після оператора  $A_4$  в ЛСА записаний кінцевий оператор, тому будуємо відповідний прямокутник, і з виходу оператора  $A_4$  будуємо дугу, яку подаємо на вхід кінцевого оператора.

В результаті вищенаведених операцій одержимо ГСА, зображену на рис. 2.15.

### **Завдання з зірочкою)))**

За поданими ЛСА побудувати ГСА:  $\omega_1 \uparrow^4 \downarrow^3 x_2 \uparrow^5 \omega_2 \uparrow^6 \downarrow^4 \bar{x}_4 \uparrow^7$

$$1 \quad A_5 A_6 \downarrow^2 \downarrow^5 A_7 A_k$$

$$A_0 \downarrow^2 A_1 x_1 \uparrow^1 x_2 \uparrow^2 \downarrow^1 A_2 x_3 \uparrow^3 \omega_1 \uparrow^4 \downarrow^3 x_1 \uparrow^5 x_2 \uparrow^6 \omega_2 \uparrow^7 \downarrow^4 A_3 \downarrow^5 x_4 \uparrow^8 \downarrow^6$$

$$2 \quad A_4 \downarrow^7 \downarrow^8 A_5 x_1 \uparrow^9 A_6 \downarrow^9 A_7 A_k$$

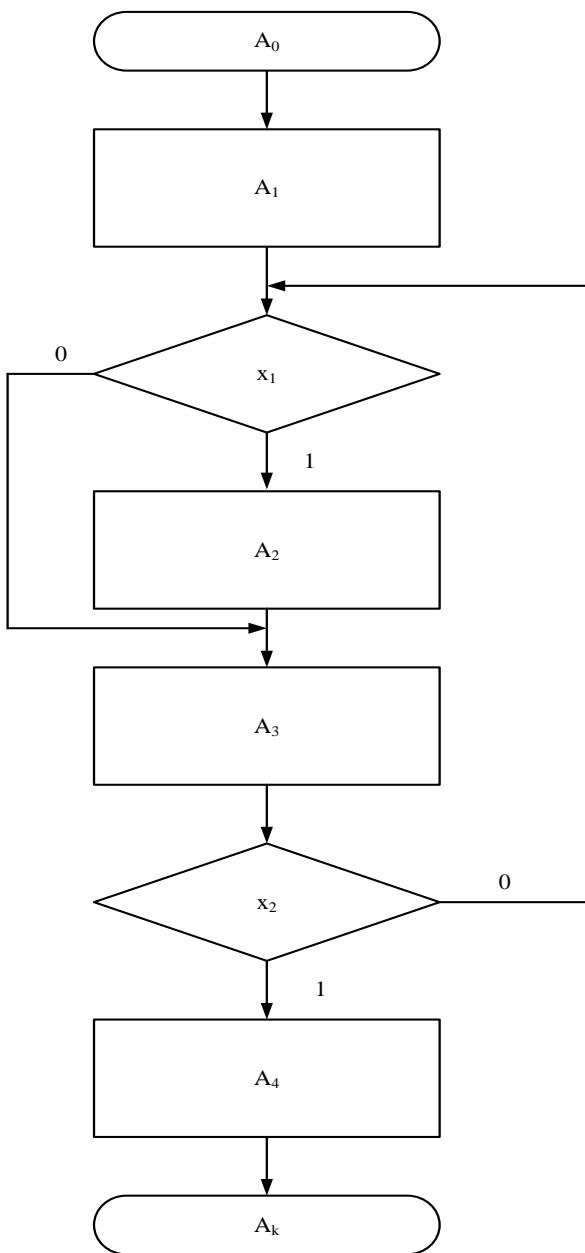


Рисунок 2.15 – Граф-схема алгоритму, побудована за заданою ЛСА



## Лекція 9. Подання роботи керуючих автоматів за допомогою матричних схем алгоритмів

Подати роботу керуючих автоматів у вигляді логічних рівнянь дозволяє використання матричних схем алгоритмів (МСА).

**Означення.** *МСА* (матрична схема алгоритму) - це квадратна матриця, рядки якої позначені символами керуючих операторів від нульового до передостаннього, а стовпці - символами операторів від першого до останнього. Елементами  $\alpha_{ij}$  матриці, де  $i$  приймає значення  $0, 1, 2, \dots, k-1$ ,  $j = 1, 2, \dots, k$  є логічні функції переходу від операторів, що відповідають рядку до операторів, що відповідають стовбцям:

	$A_1$	$A_2$	$\dots$	$\dots$	$A_k$
$A_0$	$\alpha_{01}$	$\alpha_{02}$	$\dots$	$\dots$	$\alpha_{0k}$
$A_1$	$\alpha_{11}$	$\alpha_{12}$	$\dots$	$\dots$	$\alpha_{1k}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$A_{k-1}$	$\alpha_{(k-1)1}$	$\alpha_{(k-1)2}$	$\dots$	$\dots$	$\alpha_{(k-1)k}$

Елементи  $\alpha_{ij} \in \{0, 1\}$  матриці, де  $i$  приймає значення  $i = 0, 1, 2, \dots, k-1$ ,  $j = 1, 2, \dots, k$  є кон'юнкціями однієї або декількох змінних із запереченням або без них, що відображають значення відповідної кількості логічних операторів.

**Означення.** Елементи  $\alpha_{ij} \in \{0, 1\}$ , що задають переходи від операторів рядка до операторів стовпця, називаються *логічними функціями переходів*.

Якщо функція переходу  $\alpha_{ij}$ ,  $i = 0, 1, 2, \dots, k-1$ ,  $j = 1, 2, \dots, k$  тотожно дорівнює нулю, то це означає, що відповідна їй пара операторів  $A_i$  і  $A_j$  ніколи не виконується один за одним. Для простоти заповнення МСА нульові функції переходів в матрицю можуть не заноситися.

Якщо логічна функція переходу тотожно дорівнює одиниці, то це означає, що після оператора  $A_i$  завжди виконується оператор  $A_j$ . У матриці в цьому випадку ставиться одиниця.

Якщо який-небудь керуючий оператор  $A_j$  виконується після керуючого оператора  $A_i$  при певних значеннях логічних умов, то у відповідній клітині МСА (на перетині рядка з оператором  $A_i$  і стовпця з оператором  $A_j$ ) проставляється кон'юнкція логічних умов при певних значеннях яких здійснюється перехід. При цьому символ логічної умови записується в кон'юнкцію в прямому вигляді  $x_i$ , якщо перехід здійснюється при одиничному значенні, і в інверсному  $\overline{x_i}$ , якщо перехід здійснюється при нульовому значенні. Якщо перехід від оператора  $A_i$  до оператора  $A_j$  виконується при декількох можливих комбінаціях значень логічних умов, то відповідна функція переходу є диз'юнкцією кон'юнкцій логічних умов і їх заперечень, кожна з яких відповідає одній з логічних умов переходів від оператора  $A_i$  до оператора  $A_j$ .

**Приклад.** Розглянемо МСА керуючого автомата:

	$Y_1$	$Y_2$	$Y_3$	$Y_k$
$Y_0$	$x_1$	$\overline{x_1} \overline{x_2}$	$\overline{x_1} x_2$	
$Y_1$				1
$Y_2$				1
$Y_3$				1

Звернемо увагу на те, що у розглянутому прикладі для рядка  $Y_0$  є логічна функція переходів, яка містить по дві логічні змінні.

На практиці таких змінних може бути значно більше за кількістю логічних умов, і тоді логічна функція переходів буде містити кількість змінних більше двох.

Таким чином, в  $i$ -му рядку може бути розміщено стільки логічних функцій переходу, скільки є логічних операторів від  $i$ -го керуючого оператора до інших, у тому числі може бути і до себе самото.

Побудовану матрицю необхідно перевірити на несуперечливість. Повинні виконуватися дві умови:

- Логічний добуток двох різних функцій переходу в рядку завжди має дорівнювати нулю:  

$$\alpha_{ij} \wedge \alpha_{il} = 0;$$

- Логічна сума всіх  $k$  - операторів в рядку має дорівнювати одиниці:  

$$\bigvee_{j=1}^k \alpha_{ij} = 1$$

Перша властивість виходить із того, що після виконання будь-якого керуючого оператора  $A_i$  алгоритм може перейти до виконання тільки одного оператора  $A_j$  і не більше. Тому якщо одна функція переходу  $\alpha_{ij}=1$ , то будь-яка інша повинна дорівнювати 0.

Друга властивість визначається умовою, що після виконання алгоритмом оператора  $A_i$  обов'язково повинен виконатись хоча б один із операторів  $A_j, j = 1, 2, \dots, k$ .

Виконання першої умови за поданої МСА очевидне. Перевіримо виконання другої умови. Має сенс розглянути перший рядок:

$$\begin{aligned} x_1 \vee \overline{x_1} \overline{x_2} \vee \overline{x_1} x_2 &= \\ = x_1 \vee \overline{x_1} (\overline{x_2} \vee x_2) &= x_1 \vee \overline{x_1} (1) = x_1 \vee \overline{x_1} = 1 \end{aligned}$$

Друга умова також виконується. Тому можна сказати, що МСА є несуперечливою.

Якщо побудована матриця задовольняє обом умовам, значить матрична схема алгоритму побудована правильно.

**Означення.** Логічна сума функцій переходів одного рядка називається *логічною функцією рядка*.

Оскільки логічні функції переходів являють собою логічні добутки, то логічна функція рядка є ДНФ логічної функції.

Для розглянутого вище прикладу логічними функціями рядка будуть такі функції:

$$\begin{aligned} Y_0 &\rightarrow x_1 \vee \overline{x_1} \overline{x_2} \vee \overline{x_1} x_2 \\ Y_1 &\rightarrow 1 \\ Y_2 &\rightarrow 1 \\ Y_3 &\rightarrow 1. \end{aligned}$$

## Лекція 10. Побудова матричної схеми алгоритму за заданою граф-схемою алгоритму і навпаки

На практиці МСА широко використовується для зменшення кількості операторів у ГСА шляхом їх оптимізації. Тому спочатку здійснюється перехід від ГСА до МСА, потім за МСА проводиться мінімізація кількості операторів у ГСА без змінення функцій поданого в ГСА алгоритму і тоді здійснюється перехід до нової ГСА з меншою кількістю операторів.

### 1. Побудова ГСА за МСА:

1. Зображується початкова вершина ГСА, в якій проставляється оператор початку  $A_0, i = 0$ .

2. Проводиться аналіз функцій переходів  $\alpha_{ij} = \alpha_{0j}$ , записаних у відповідному оператору  $A_0$  рядку матриці. Якщо одна з цих функцій дорівнює 1, то інші дорівнюють 0. Це означає, що за оператором початку  $A_0$  безпосередньо йде другий керуючий оператор  $A_1$ , який стоїть у стовпці, що відповідає 1.

Очевидно, що дві і більше одиниці не повинні стояти в одному рядку, оскільки в протилежному разі буде порушена перша умова МСА:

$$\alpha_{ij} \wedge \alpha_{il} = 0, j \neq l.$$

Також не повинні дорівнювати нулю усі  $\alpha_{ij}$ , одного рядка. Якщо це буде не так, то тоді порушиться друга умова МСА:

$$\bigvee_{j=1}^k \alpha_{ij} = 1$$

3. Будується вершина для оператора  $A_j$ , в яку проставляється її символічне позначення, і проводиться з'єднання її входу із виходом оператора  $A_0$ .

4. У випадку, якщо в одній із клітинок рядка оператора  $A_0$  функцією переходів є змінна  $x_\gamma, \gamma = 1, 2, \dots, p$ , то будується вершина умовного оператора, яка має один вхід. Він об'єднується

із виходом оператора  $A_0$ , і два виходи, які об'єднуються з операторами, що стоять у стовпцях, відповідних змінних  $x_\gamma$  і  $\overline{x_\gamma}$ . Змінна  $\overline{x_\gamma}$  обов'язково повинна бути наявна у рядку, у якому знаходиться змінна  $x_\gamma$ , і навпаки. Якщо ця вимога буде порушена, то в результаті може не виконуватись перша умова МСА:

$$\alpha_{ij}\alpha_{il} = 0, j \neq l.$$

5. Якщо у рядку є функції переходів  $\alpha_{ij}$ , які містять добутки двох і більше змінних  $x_\gamma$ , то це означає, що при виконанні першої і другої умов МСА вихід вершини  $A_i$  з'єднується через логічні вершини з кількістю операторів, відповідних кількості функцій переходів, що містять логічні змінні  $x_\gamma$ . Ці оператори знаходяться у стовпцях, відповідних клітинкам рядка, в яких знаходяться зазначені функції переходів.

6: Після побудови усіх вершин і зв'язків, відповідних рядку оператора  $A_i$  відбувається перехід до рядка  $A_{i+1}$  і виконуються аналогічні операції з подальшої побудови ГСА, що і для рядка  $A_i$ .

Потім відбувається побудова ГСА стосовно рядка  $A_{i+1}$  і т.д. до  $A_k$ .

## **2. Побудова МСА за ГСА:**

1. У результаті аналізу ГСА складається квадратна матриця МСА. Для цього рядки матриці позначаються символами операторних вершин за винятком оператора  $A_k$  «кінець». Перший верхній рядок матриці позначається символом  $A_0$  «початок». Стовпці матриці позначаються символами операторних вершин, починаючи з оператора  $A_1$ . Крайній стовпець позначається символом операторної вершини  $A_k$ .

2. Для знайдення функцій переходів між операторною вершиною  $A_i$  та усіма іншими операторними вершинами ГСА визначаються усі можливі шляхи, які об'єднують вершину  $A_i$  з іншими операторними вершинами ГСА, в тому числі і які проходять через логічні (умовні) вершини.

3. Для кожного із знайдених  $x$  шляхів складається функція переходу, що діє за такими правилами:

а) якщо вершина  $A_i$  і деяка інша  $A_j$  з'єднані безпосередньо одна з одною, то відповідна їм функція переходів тотожно дорівнює 1, тобто  $\alpha_{ij} = 1$ ;

б) якщо шлях між вершиною оператора  $A_i$  і деякою іншою  $A_j$  проходить через 1(0) вихід в умовній вершині, то у функцію переходу входить символ умовної вершини у прямому або інверсному вигляді, тобто  $\alpha_{ij} = x_k$  або  $\alpha_{ij} = \overline{x_k}$ ;

в) якщо шлях, який об'єднує вершину  $A_i$  з вершиною  $A_j$ , проходить через декілька умовних вершин, то відповідні функції переходів утворюють кон'юнкцію, наприклад,

$$\alpha_{ij} = \overline{x_k} x_l x_m$$

г) якщо між вершиною  $A_i$  і деякою вершиною  $A_j$  відсутній шлях, який не проходить також і через інші умовні вершини, то відповідна функція переходів  $\alpha_{ij}$  тотожно дорівнює 0.

4. Після одержання МСА необхідно упевнитись, що виконуються перша і друга умови МСА:

$$\alpha_{ij}\alpha_{il} = 0,$$

де  $j \neq l, i$

$$\bigvee_{j=1}^k \alpha_{ij} = 1$$

## Лекція 11. Перетворення логічної схеми алгоритму в матричну схему алгоритму і навпаки

Звичайно алгоритм керування вволиться і зберігається в пам'яті керуючої ЕЦОМ у вигляді ЛСА. Потім за необхідності відбувається перехід від ЛСА до МСА, яка мінімізується за кількістю операторів, і потім відбувається зворотний перехід від МСА до ЛСА. Тому важливо знати алгоритми перетворення ЛСА в МСА для того, щоб за необхідності розробити відповідні програми переходу для керуючих ЕЦОМ.

### 1. Побудова МСА за ЛСА

1. Внаслідок аналізу ЛСА визначаються усі оператори  $A_0, A_1, \dots, A_k$ , що входять до неї, після чого будується матриця, рядки якої позначаються операторами  $A_0, A_1, \dots, A_{k-1}$ , а стовпці операторами  $A_1, A_2, \dots, A_k$ .

2. Розглядаються оператори ЛСА  $A_i$ , починаючи з  $A_0$ , і визначаються усі оператори, які можуть бути виконані після  $A_i$  за певних логічних умов.

3. Складаються функції переходів між оператором  $A_i$  й усіма іншими операторами за такими правилами:

а) якщо деякий оператор  $A_j$  виконується безпосередньо після  $A_i$  незалежно від значень яких-небудь логічних умов, то відповідна функція тотожно дорівнює 1;

б) якщо деякий оператор  $A_j$  ЛСА, що розглядається, ні за яких значень логічних умов не виконується після оператора  $A_i$ , то відповідна функція переходів тотожно дорівнює нулю -  $\alpha_{ij} = 0$ ;

в) якщо деякий оператор  $A_j$  виконується після  $A_i$  за певною комбінацією значень логічних умов, то відповідна функція переходів записується як кон'юнкція тих логічних умов, за певних значень яких здійснюється перехід до керуючого оператора, при цьому символ логічної умови записується у кон'юнкції в прямому вигляді, якщо перехід здійснюється при одиничному значенні логічної умови, і в інверсному, якщо перехід здійснюється за нульовим значенням логічної умови.

4. Функції переходів, отримані відповідно до пункту 3, записуються в рядок матриці, відповідний оператору  $A_i$ .

5. Дії за пп. 2, 3, 4 послідовно виконуються для всіх операторів  $A_i$ , які входять в ЛСА, що розглядається, за винятком оператора  $A_k$ .

6. Здійснюється перевірка побудованої МСА на відповідність її першій і другій логічним умовам для МСА:

$$\alpha_{ij}\alpha_{il} = 0,$$

де  $j \neq l, i$

$$\bigvee_{j=1}^k \alpha_{ij} = 1$$

Для цього складаються логічні функції рядків і для кожної з них здійснюється перевірка на відповідність логічним умовам.

## 2. Побудова ЛСА за МСА

1. Для кожного рядка МСА формуються *формули перетворень* у вигляді суми кон'юнкцій, утворених із операторів і відповідних їм функцій переходів:

$$A_i \rightarrow \alpha_{i1}A_1 + \alpha_{i2}A_2 + \dots + \alpha_{ij}A_j + \dots + \alpha_{ik}A_k.$$

2. Із системи формул перетворення виділяється формула, відповідна оператору  $A_i$ , починаючи з  $A_0$ :

$$A_0 \rightarrow \alpha_{01}A_1 + \alpha_{02}A_2 + \dots + \alpha_{0j}A_j + \dots + \alpha_{0k}A_k.$$

За цією формулою визначаються можливі зв'язки оператора  $A_0$  з іншими  $A_1, A_2, \dots, A_k$ :

а) якщо одна з функцій переходів  $\alpha_{ij} = 1$ , а інші дорівнюють нулю, то в цьому випадку формула перетворення має



вигляд  $A_i \rightarrow A_j$ , це значить, що після оператора  $A_i$  йде оператор  $A_j$ ;

b) якщо декілька функцій переходів формули перетворення подаються логічними добутками, то для подальшого запису ЛСА аналізується перший зліва член (кон'юнкція) формули перетворення, в якому знаходиться керуючий оператор  $A_i$ , що не зустрічався раніше в ЛСА, яка будується, і записуються всі логічні змінні цього члена із запереченням або без, кожна з яких праворуч забезпечується початком пронумерованої стрілки, і потім записується оператор  $A_i$ , що входить в цей член формули.

3. Далі відбувається звернення до формули перетворення  $A_{i+1}$  і для неї повторюється процедура за п. 2.

4. Зазначена процедура виконується доти, поки всі оператори і логічні умови не увійдуть в ЛСА, що формується.

5. У кожній з формул перетворень шукаємо у функціях переходів змінні, інверсні відносно тих, що вже є в ЛСА. Ставимо перед операторами, які йдуть за ними і повинні бути в ЛСА, кінці: стрілок.

Щоб переконатися в правильності переходу від МСА до ЛСА необхідно виконати зворотне перетворення від ЛСА до МСА.

### **Завдання з зірочкою)))**

1. За поданою МСА записати ЛСА та ГСА:

	$A_1$	$A_2$	$A_3$	$A_4$	$A_5$	$A_k$
$A_0$	1					
$A_1$	$\bar{x}_1 \bar{x}_2$	$x_1$		$\bar{x}_1 x_2$		
$A_2$			1			
$A_3$	1					
$A_4$					1	
$A_5$	$\bar{x}_2 x_3$			$x_2 x_3$		$\bar{x}_3$

## Лекція 12. Мінімізація матричної схеми алгоритму за кількістю логічних умов

У моделях МПА, поданих у вигляді ГСА або ЛСА, можуть бути декілька однакових операторів або логічних умов.

Для зменшення їх кількості проводиться перетворення цих моделей в МСА, і потім виконується їх мінімізація за наступними правилами:

1. Будується МСА і перевіряється на несуперечність.
2. За МСА складається система формул перетворень.
3. У кожній формулі перетворення знаходять змінні, які входять в усі члени формул.
4. Шукають змінні, які є спільними для більшості формул перетворення.
5. Ці змінні виносять за дужки таким чином, щоб одержати вираз типу  $(x_i A_k \vee x_i A_l)$ . Такі вирази називають *елементарними дужками*. Оскільки в МСА може бути декілька варіантів винесення за дужки, то отримують систему дужкових виразів.
6. Аналізується отримана система дужкових виразів. Якщо є однакові вирази в елементарних дужках, то залишають тільки один, той що зустрівся першим. Інші такі ж вирази замінюють тотожними логічними умовами  $\omega \uparrow^i$ , кінці стрілок від яких займають місце перед виразом, що залишився у дужках, в одній із формул перетворення.
7. За одержаними формулами перетворення будується еквівалентна мінімальна ЛСА або ГСА.

**Приклад.** Початкова ЛСА має вигляд:

$$A_0 A_1 x_1 \uparrow^1 \downarrow^3 \downarrow^6 A_2 x_2 \uparrow^2 \omega_1 \uparrow^4 \downarrow^2 x_1 \uparrow^5 \omega_2 \uparrow^6 \downarrow^4 \bar{x}_3 \uparrow^3 A_3 \downarrow^1 \downarrow^5 A_4 A_k$$

Необхідно провести мінімізацію МСА за кількістю логічних умов.

**Розв'язування.** Умова функціонування автомата задана у вигляді ЛСА, оскільки це найбільш компактна форма подання алгоритму. Виконаємо перехід до ГСА, а потім до МСА.

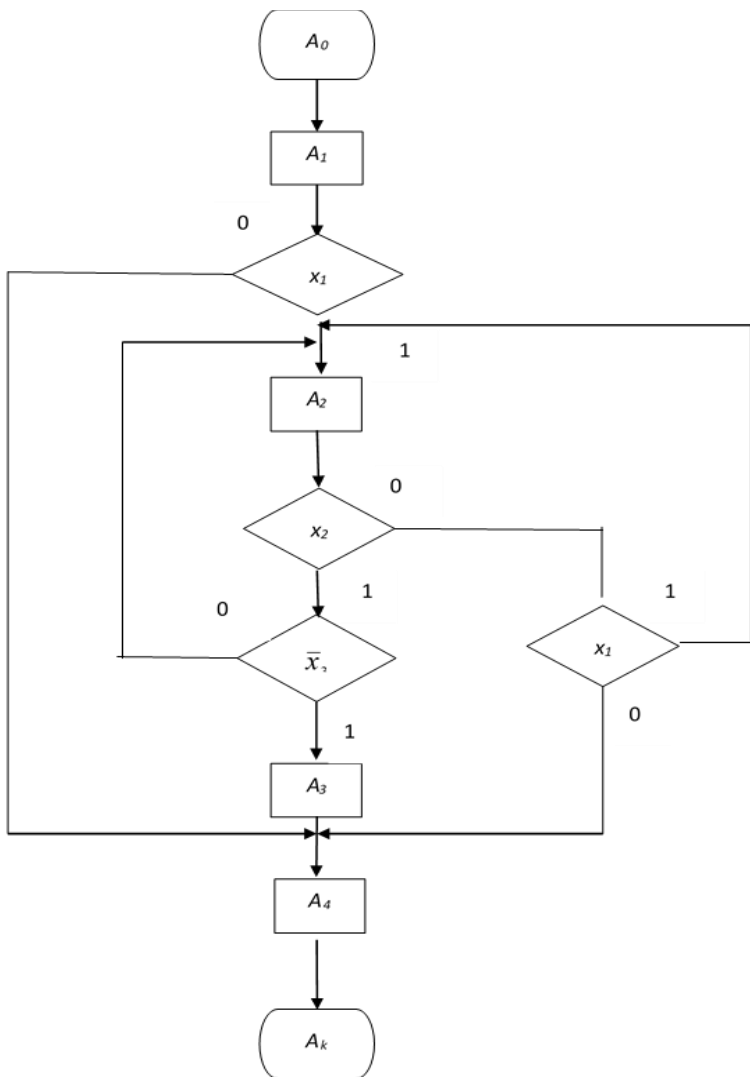


Рисунок 2.16 – Початкова ГСА

За ГСА побудуємо матричну схему та перевіримо її на несуперечність:

	$A_1$	$A_2$	$A_3$	$A_4$	$A_k$
$A_0$	1				
$A_1$		$x_1$		$\bar{x}_1$	
$A_2$		$x_2 x_3 \vee \bar{x}_2 x_1$	$x_2 \bar{x}_3$	$\bar{x}_2 \bar{x}_1$	
$A_3$				1	
$A_4$					1

За побудованою МСА запишемо систему формул перетворень, винесемо однакові співмножники, сформуємо елементарні дужки:

$$\begin{aligned}
 A_0 &\rightarrow A_1; \\
 A_1 &\rightarrow x_1 A_2 \vee \bar{x}_1 A_4 = \downarrow x_1 A_2 \vee \bar{x}_1 A_4 \\
 A_2 &\rightarrow x_2 x_3 A_2 \vee \bar{x}_2 x_1 A_2 \vee x_2 \bar{x}_3 A_3 \vee \bar{x}_2 \bar{x}_1 A_4 = \\
 &= x_2 (x_3 A_2 \vee \bar{x}_3 A_3) \vee \bar{x}_2 (x_1 A_2 \vee \bar{x}_1 A_4) = \\
 &= x_2 (x_3 A_2 \vee \bar{x}_3 A_3) \vee \bar{x}_2 \omega \uparrow; \\
 A_3 &\rightarrow A_4; \\
 A_4 &\rightarrow A_k.
 \end{aligned}$$

За системою побудуємо ГСА з урахуванням проведених спрощень. Нова ГСА має вигляд, представлений на рисунку 2.

ЛСА, що відповідає спрощеній ГСА має наступний вигляд:

$$A_0 A_1 \downarrow^2 x_1 \uparrow^1 \downarrow^3 A_2 x_2 \uparrow^2 \bar{x}_3 \uparrow^3 A_3 \downarrow^1 A_4 A_k.$$

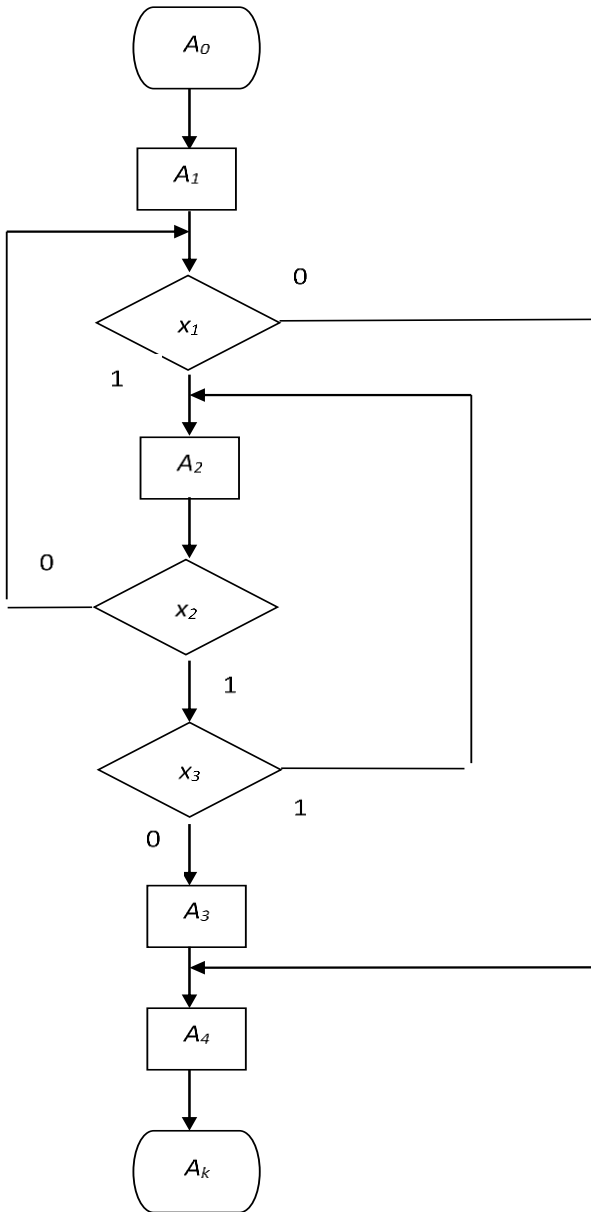


Рисунок 2.17 – Нова ГСА, побудована з урахуванням мінімізації алгоритму за кількістю логічних умов

### *Завдання з зірочкою)))*

1. За заданою ЛСА побудувати ГСА та МСА. Записавши систему формул перетворень, провести мінімізацію МСА, отримати мінімальну ЛСА та ГСА:

$$A_0 A_1 x_1 \uparrow^1 \downarrow^5 A_2 x_2 \uparrow^2 \omega \uparrow^3 \downarrow^1 \bar{x}_2 \uparrow^4 \downarrow^2 A_3 \downarrow^6 A_4 \omega \uparrow^5 \downarrow^3 \downarrow^4 A_5 \bar{x}_3 \uparrow^6 A_6 A_\kappa$$

2. За заданою ЛСА побудувати ГСА та МСА. Записавши систему формул перетворень, провести мінімізацію МСА, отримати мінімальну ЛСА та ГСА:

$$A_0 A_1 x_1 \uparrow^1 \downarrow^5 A_2 \omega \uparrow^2 \downarrow^1 \downarrow^4 A_3 \downarrow^2 A_4 \bar{x}_2 \uparrow^3 x_1 \uparrow^4 \omega \uparrow^5 \downarrow^3 A_\kappa$$

3. За заданою ЛСА побудувати ГСА та МСА. Записавши систему формул перетворень, провести мінімізацію МСА, отримати мінімальну ЛСА та ГСА:

$$A_0 A_1 A_2 x_1 \uparrow^1 A_3 x_2 \uparrow^2 x_3 \uparrow^3 \omega \uparrow^4 \downarrow^1 \bar{x}_3 \uparrow^5 \downarrow^3 A_4 \downarrow^2 A_5 \downarrow^4 \downarrow^5 A_6 A_\kappa$$

### Розділ III. Керуючі автомати зі схемною логікою

#### Лекція 13. Керуючі автомати на комбінаційних схемах

На практиці широко використовуються схемні рішення для побудови простих пристроїв керування. При цьому для характеристики станів об'єкта, що керується, використовуються набори двійкових змінних, які одержуються від датчиків, перемикачів на пульті керування, кінцевих перемикачів і т.ін. При занесенні в пристрій керування значень цих змінних вони об'єднуються в набори, які відповідають тим чи іншим ситуаціям в керуючому об'єкті. У відповідності до цих наборів виробляються команди ввімкнути (1) або вимкнути (0). Якщо в процесі розв'язання задачі виникає необхідність його розгалуження, то використовують системи логічних функцій, в яких вводиться в дію та або інша логічна функція. Зручною формою їх задання є заснована на таблиці істинності таблиця рішень (див. табл. 3.1). У ній набори двійкових змінних являють собою логічні умови, залежно від яких виробляється та чи інша керуюча команда 1 або 0, що в кінцевому підсумку приводить до виконання або невиконання  $i$ -го набору значень логічних умов ( $i = 1, 2, \dots, n$ ).

Таблиця 3.1 – Таблиця рішень

Номер набору умов	Умова				Вектор Дій (рішення)
	$x_1$	$x_2$	...	$x_n$	
1	-	0	...	1	0
2	1	1	...	-	1
.	.	.	...	.	.
.	.	.	...	.	.
.	.	.	...	.	.
$n$	0	1	...	1	1

Набори умов, які називають ще *правилами*, можуть мати прочерки для тих умов, які не є суттєвими. Замість прочерків може бути поставлено 1 або 0, але це ніяк не повинно впливати на результуючі дії.

Сукупність значень дій на всіх наборах змінних (умовах) має назву *вектора дій* або *рішення*. Цих векторів може бути декілька, а отже, і декілька логічних функцій, які створюють їх систему.

Існують різні шляхи реалізації таблиці рішень - апаратний, мікропрограмний та програмний.

Для нескладних задач керування найбільш ефективним та швидкодіючим способом реалізації буде апаратний у вигляді схеми або на основі ПЗП, в яке записується таблиця рішень. Набори змінних при цьому відповідають адресам, а значення вектора дій - складу відповідних комірок пам'яті.

**Приклад 1.** Розробити пристрій керування для сортування деталей розміром  $b$ ,  $2b$  та  $3b$ , які розміщені поперек роликів транспортеру. Пристрій керування має на основі сигналів датчиків  $d_1$ ,  $d_2$  та  $d_3$ , що також розміщені поперек транспортеру, визначити розмір деталі та провести її сортування у відповідні нагромаджувачі деталей з заслінками  $B_1$ ,  $B_2$  та  $B_3$ , призначених для відбирання деталей кожного типу. Контроль розмірів деталей здійснюється датчиками  $d_1$ ,  $d_2$  та  $d_3$ .

Пристрій для сортування деталей за розміром наведено на рисунку 3.1.

**Розв'язування.** Можливі наступні чотири ситуації:

а) на транспортері знаходиться деталь найменшого розміру  $b$  – перекритий  $d_1$ ,  $d_2$  або  $d_3$ , і відкривається бункер  $B_1$ ;

б) на транспортері знаходиться деталь середнього розміру  $2b$  – перекриті два датчики  $d_1$ ,  $d_2$  або  $d_2$ ,  $d_3$  і відповідно відкривається бункер  $B_2$ ;

в) на транспортері деталь розміру  $3b$  – перекриті всі три датчики і відкритий бункер  $B_3$ .

г) у випадку, коли спрацьовують датчики  $d_1$  і  $d_3$  та відсутній сигнал від датчика  $d_2$ , виробляється сигнал – «несправність датчика  $d_2$ ».

Після аналізу умов функціонування керуючого пристрою складають ГСА роботи керуючого автомата. На рисунку 3.2 наведена змістовна ГСА, а на рисунку 3.3 – відповідна символна ГСА.



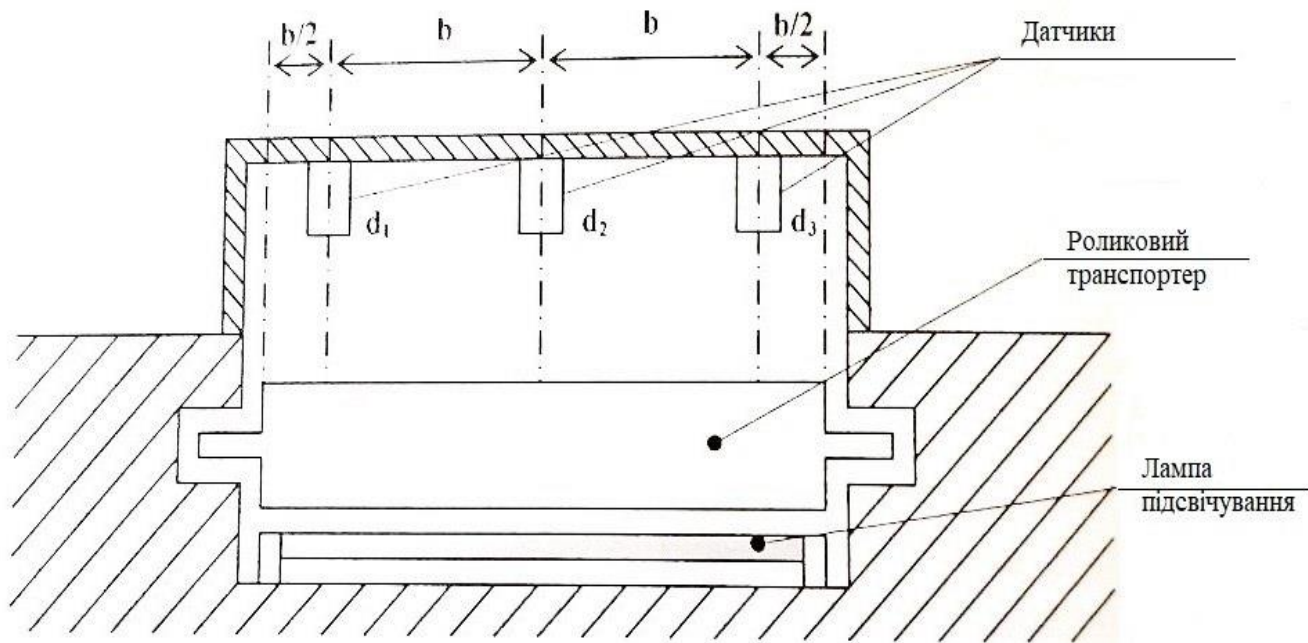


Рисунок 3.1 – Пристрій для сортування деталей за розміром



Рисунок 3.2 – Змістовна ГСА роботи керуючого автомата сортування деталей

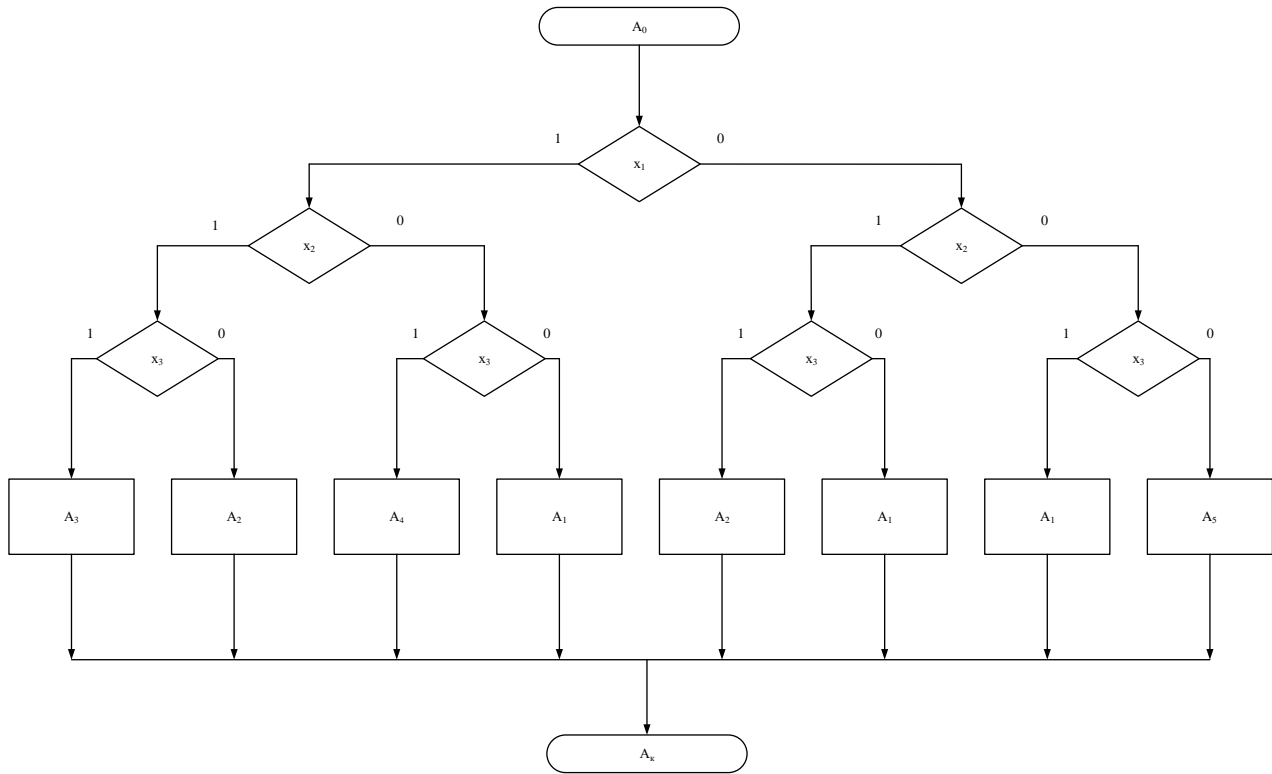


Рисунок 3.3 - Символьна ГСА керуючого автомата сортування деталей

За символною ГСА складають таблицю рішень керуючого автомата – таблиця 3.2.

Таблиця 3.2 - Таблиця рішень керуючого автомата сортування деталей

Номер набору умов	Умова			Вектор дій				
	$x_1$	$x_2$	$x_3$	$A_1$	$A_2$	$A_3$	$A_4$	$A_5$
0	0	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0
2	0	1	0	1	0	0	0	0
3	0	1	1	0	1	0	0	0
4	1	0	0	1	0	0	0	0
5	1	0	1	0	0	0	1	0
6	1	1	0	0	1	0	0	0
7	1	1	1	0	0	1	0	0

За таблицею рішень запишемо ДДНФ логічних функцій, що складають вектор рішення:

$$A_1 = \overline{x_1} \overline{x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2} \overline{x_3}$$

$$A_2 = \overline{x_1} x_2 x_3 \vee x_1 x_2 \overline{x_3}$$

$$A_3 = x_1 x_2 x_3$$

$$A_4 = x_1 \overline{x_2} x_3$$

$$A_5 = \overline{x_1} \overline{x_2} \overline{x_3}$$

На наступному кроці виконують мінімізацію одержаних формул, переходять до заданого функціонального базису, будують та. досліджують схему.

Таким чином, виконано завдання синтезу керуючого автомату сортування деталей за розміром.

## Розділ IV. Мікропрограмні автомати

### Лекція 14. Найпростіші мікропрограмні автомати

#### *Історична довідка:*

- Подальший розвиток принципу «розділяй і володарюй» у дискретній техніці призвів до створення МПА - мікропрограмного автомата (1951, Кембрідж, **Моріс Вінсент Вілкс** ).
- Цифрові автомати, що використовують для своєї роботи мікропрограми, називаються *мікропрограмними автоматами* (МПА).



*Моріс Вінсент Вілкс* (англ. *Maurice Vincent Wilkes*, 26 червня 1913 – 29 листопада 2010, Дадлі, Великобританія) - британський учений у галузі комп'ютерних наук.

Професор Вілкс найбільш відомий як проектувальник EDSAC (Англ. Electronic delay storage automatic calculator – автоматичний обчислювач на електронних лініях затримки). - першого комп'ютера, що припускає внутрішнє зберігання програм. Побудований у 1949, EDSAC використовував пам'ять на лініях затримки. Покоління EOM EDVAC і EDSAC завершили перехід у конструкціях обчислювальних машин від пристроїв на основі електричних реле до пристроїв на основі електронних ламп.

Моріс Вілкс також відомий у співавторстві з Віллером і Гіллом як автор книги «Preparation of programs for electronic digital computers», 1951 року, в якій вводиться найважливіше поняття бібліотеки програм.

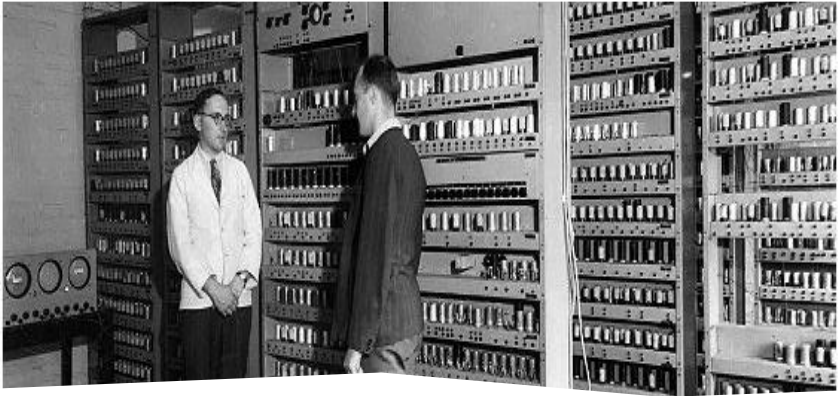


Рисунок 4.1 - Моріс Вілкс і Білл Ренвік зі своєю EDSAC

### ***Основні наукові здобутки Моріса Вінсента Вілкса***

- Здобув освіту в коледжі короля Едуарда VI у Сторбриджі і в шкільні роки займався радіоаматорством, до якого його залучив вчитель хімії.
- Навчався з 1931 по 1934 рік в Кембриджському університеті, де у 1936 році отримав звання доктора філософії, написавши дисертацію на тему: «Радіопоширення дуже довгих радіохвиль в іоносфері».
- У 1945 році Вілкс був призначений другим директором Математичної лабораторії Кембриджського університету, пізніше відому як Комп'ютерна лабораторія (англ. *Computer Laboratory*).
- У серпні 1946 Вілкс відвідує США. Після повернення до Кембриджу Вілкс приступає до створення EDSAC (Автоматичний обчислювач на електронних лініях затримки). Він використовував тільки перевірені методи побудови кожної частини обчислювачів. У результаті обчислювач був повільнішим і меншим, ніж інші його сучасники. Однак, лабораторний комп'ютер його був першим, який зберігав програмний код, і успішно працював з травня 1949 року.
- У 1951 році він розробив концепцію **мікропрограмування**, яка була реалізована тим, що

центральный процессор в комп'ютері міг знаходитися під контролем мініатюрної, високо спеціалізованої комп'ютерної програми, що містилася на високошвидкісних дисках. Це поняття значно спрощувало розвиток процесорів.

- Вілксу також приписують ідею символічних міток, **макросів** і бібліотечних підпрограм. Ці фундаментальні зміни зробили програмування набагато простішим і проклали шлях для мов програмування високого рівня.
- Пізніше Вілкс працював над ранніми системами поділу часу (які тепер називаються багатокористувацькими операційними системами) та розподілених обчислень.
- У 1974 році Уілкс зіткнувся з швейцарською мережею передачі даних (у Hasler AG), яка використовувала кільцеву топологію для розподілу часу в мережі. Завдяки Уілксу були сформовані комерційні товариства, і аналогічні технології стали широко доступними у Великобританії.

### ***Принцип мікропрограмування («розділяй і володарюй»)***

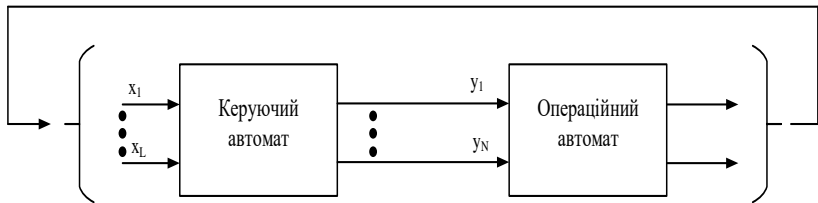


Рисунок 4.2 – Ілюстрація принципу «Розділяй і володарюй»

- Мікрооперації - МО - елементарні дії обробки інформації.
- Мікрокоманди -МК - набір МО, що виконуються в одному такті.
- Мікропрограма – послідовність МК. Таким чином, мікропрограма (англ. *firmware*, "прошивка") - програмне забезпечення, вбудоване ("зашите") в апаратний пристрій. Часто є мікросхемою флеш-ПЗП чи у вигляді файлів

образів мікропрограми, які можуть бути завантажені в апаратне забезпечення.

- Далі – Команди та Програми тощо.

Крім схем з жорсткою логікою для вирішення задачі керування широко застосовуються мікропрограмні автомати різної міри універсальності та складності. Але загальним для них є наявність в їх структурі керуючої програми.

Найпростіші керуючі автомати, що програмуються, ґрунтуються на простому переборі керуючих команд. У цьому випадку кожна подальша команда  $A_{i+1}$  виконується після закінчення попередньої  $A_i$ ,  $i=1, 2, \dots, n$ , що забезпечується асинхронною роботою автомата, коли на його тактований вхід надходять сигнали від кінцевих перемикачів керованого ним обладнання. Граф-схема алгоритму (ГСА) такого автомата є лінійною, являючи собою тільки послідовність керуючих команд, і не має ні розгалужень, ні циклів (див. рис. 4.3).

Програма роботи автомата, що відповідає такій ГСА, найбільш проста і складається з послідовного перебору станів автомата. Ці стани, подані в двійковій формі, називаються *мікрокомандами*, а створені ними програми – *мікропрограмами*.

**Означення.** Цифрові автомати, що використовують для своєї роботи мікропрограми, називаються *мікропрограмними автоматами* (МПА).



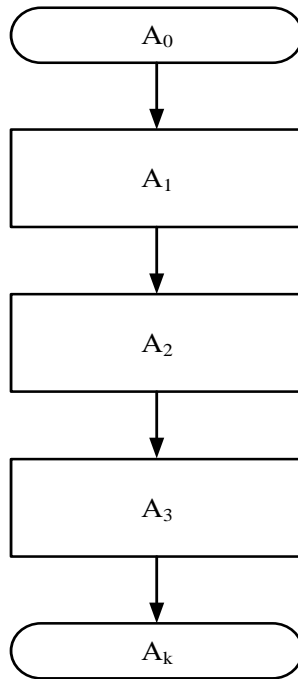


Рисунок 4.3 – Лінійна ГСА

Наведемо структуру найпростішого МПА, що реалізує лінійну ГСА: (рис. 4.4).

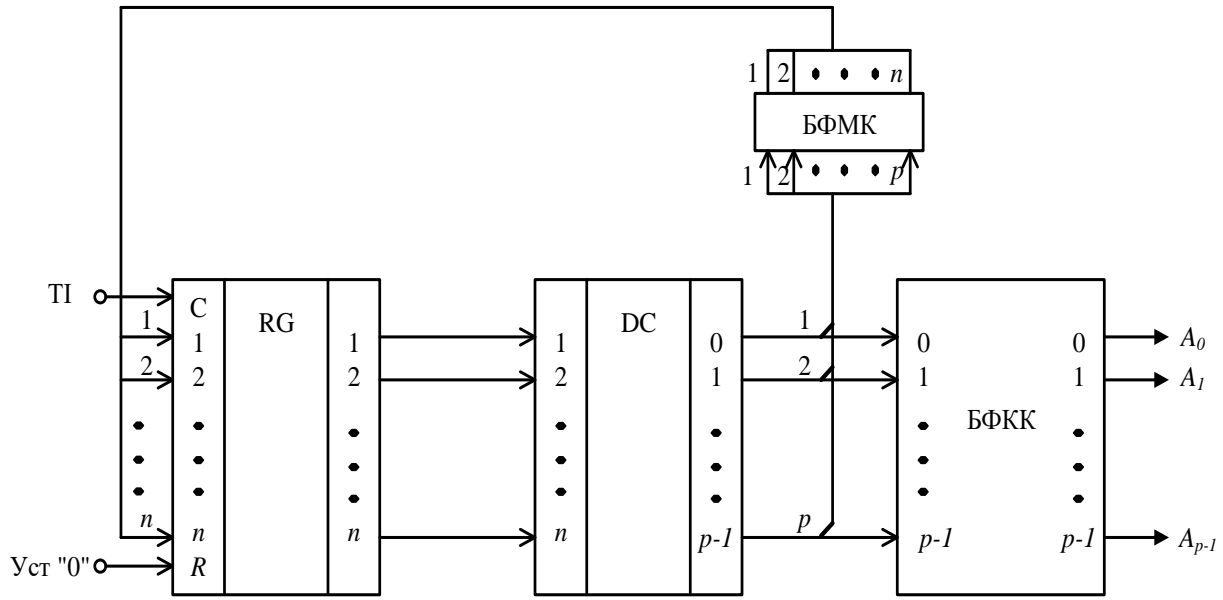


Рисунок 4.4 – Найпростіший керуючий МПА на регістрі

Найпростіший мікропрограмний автомат містить регістр RG, дешифратор DC, блок формування команд керування (БФКК) і блок формування мікро-команд (БФМК).

Блок формування команд керування вирішує завдання підсилення і формування команд керування, які надходять потім на виконавчі органи об'єкта керування.

Блок формування мікрокоманд призначений для перетворення вихідних сигналів дешифратора DC в мікрокоманди, що являють собою двійкові кодові комбінації.

Функціонально цей блок є шифратором, що перетворює сигнал на одному із своїх  $p$  входів в  $n = \lceil \log_2 p \rceil$  виходів.

Працює МПА, що розглядається, таким чином. У початковому стані за відсутності тактового імпульсу ТІ регістр RG встановлений сигналом Уст «0» в початковий нульовий стан. Відповідно на виході «0» дешифратора DC буде вироблятися сигнал, який надходить на відповідний йому вхід БФМК, де він перетворюється у двійкову кодову комбінацію мікрокоманди довжини  $n$ , яка потім подається на відповідні входи регістра RG. Одночасно із цим сигнал з дешифратора DC надходить також на відповідний йому вхід БФКК, де відбувається його формування і підсилення.

При надходженні тактового імпульсу ТІ на синхровхід С регістра RG в нього заноситься нова мікрокоманда з БФМК, яка повинна виконатися услід за початковою мікрокомандою, і потім цикл роботи МПА повторюється відповідно до вищеописаного.

Розглянувши на рис. 4.4 структуру МПА можна спростити, ввівши замість регістра лічильник, наприклад, з вісьмома станами – 000, 001, 010, 011, 100, 101, 110, 111 (див. рис. 4.5).

Тоді нова структура МПА буде містити двійковий лічильник СТ2 на три розряди, дешифратор DC із вісьмома виходами і БФКК.

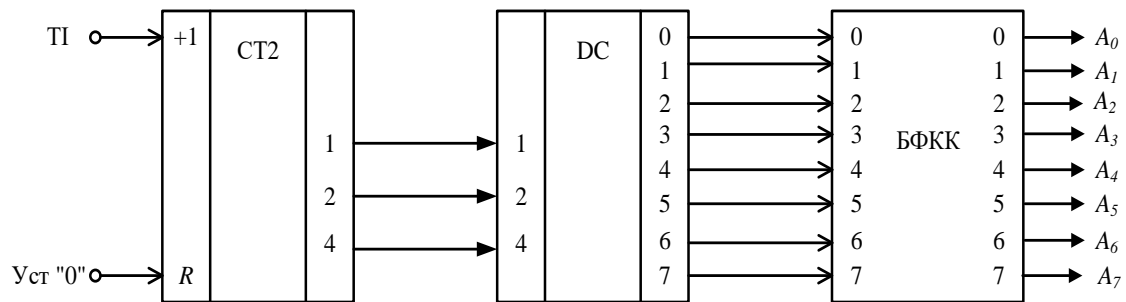


Рисунок 4.5 - Найпростіший керуючий МПА на лічильнику

У початковий нульовий стан керуючий автомат встановлюється імпульсом на вході Уст «0». За тактовими імпульсами ТІ, що надходять на лічильний вхід «+ 1», лічильник послідовно проводить» перебір своїх станів, які потім розшифровуються дешифратором ДС і подаються на входи БФКК. У результаті на виходах БФКК послідовно з'являються керуючі команди  $A_0, A_1, \dots, A_7$ , які подаються на об'єкт керування до появи кінцевого сигналу  $A_k = A_7$  про закінчення пропесу керування.

МПА можна також реалізовувати за допомогою розподільників імпульсів, які набули широкого розповсюдження в різних керуючих пристроях внаслідок простоти своєї структури і доброї надійності. Однак число тригерів, що міститься в них, дорівнює не  $\lceil \log_2 p \rceil$ , як це було в МПА з регістрами чи лічильниками, а  $p$ , що при великій кількості команд керування призводить до додаткових апаратних витрат.

Важливим показником ефективності МПА є час *перемикання* команд керування, а також час «перекриття» команд, які змінюються.

Час перекриття являє собою час, протягом якого співіснують дві команди, перша з яких виконується, а друга в цей час приходить. Процес перекриття називають ще *гонками*.

Час перемикання МПА впливає на його швидкодію, а гонки - на його надійність. Тому час перемикання і гонок намагаються зменшувати, підвищуючи таким чином швидкодію і надійність МПА. Це досягається застосуванням складних структур лічильників та розподільників імпульсів з протигончними елементами і спеціальним кодуванням.

## Лекція 15. Універсальні мікропрограмні автомати Уїлкса

Розвиваючи структуру найпростішого керуючого мікропрограмного автомата, легко перейти до універсального мікропрограмного автомата, суттєвою відміною якого від найпростішого є те, що в ньому допускаються цикли і розгалуження у мікропрограмі, яка ним виконується.

**Означення.** Автомат, алгоритм якого задається введеною в нього мікропрограмою, що містить цикли і розгалуження, називається *універсальним мікропрограмним автоматом*.

Таким автоматом є автомат Уїлкса. Мікропрограмний автомат Уїлкса складається із регістра RG, дешифратора DC, блока перевірки логічних умов (БПЛУ), блока формування команд керування (БФКК) і блока формування мікрокоманд (БФМК), який являє собою звичайний шифратор (див. рис. 4.6).

БПЛУ має дві групи входів, на одну з яких надходять сигнали з  $p$  виходів дешифратора, а на другу – логічні умови  $x_1, x_2, \dots, x_p$  від датчиків, що знаходяться на керованому об'єкті. Перевірка цих умов і вироблення на їх основі подальшого шляху виконання мікропрограми є основною функцією БПЛУ. Завдяки такій функції можливе здійснення розгалужень і циклів, що реалізуються мікропрограмами у автоматах.

Число умов може бути будь-яке, але не більше числа виходів дешифратора  $p$ . Тому максимальне число виходів в БПЛУ не може перевищувати  $2p$ .

Кожна логічна умова в автоматі Уїлкса перевіряється за допомогою логічного вузла у БПЛУ, який реалізується двома двовходовими схемами I (див. рис. 4.7), на один із двох входів яких надходить сигнал  $z(t)$  із дешифратора, що діє в момент часу  $t$ , а на інші входи цих схем - прямий і інверсний сигнали логічної умови  $x(t)$  і  $\bar{x}(t)$ .

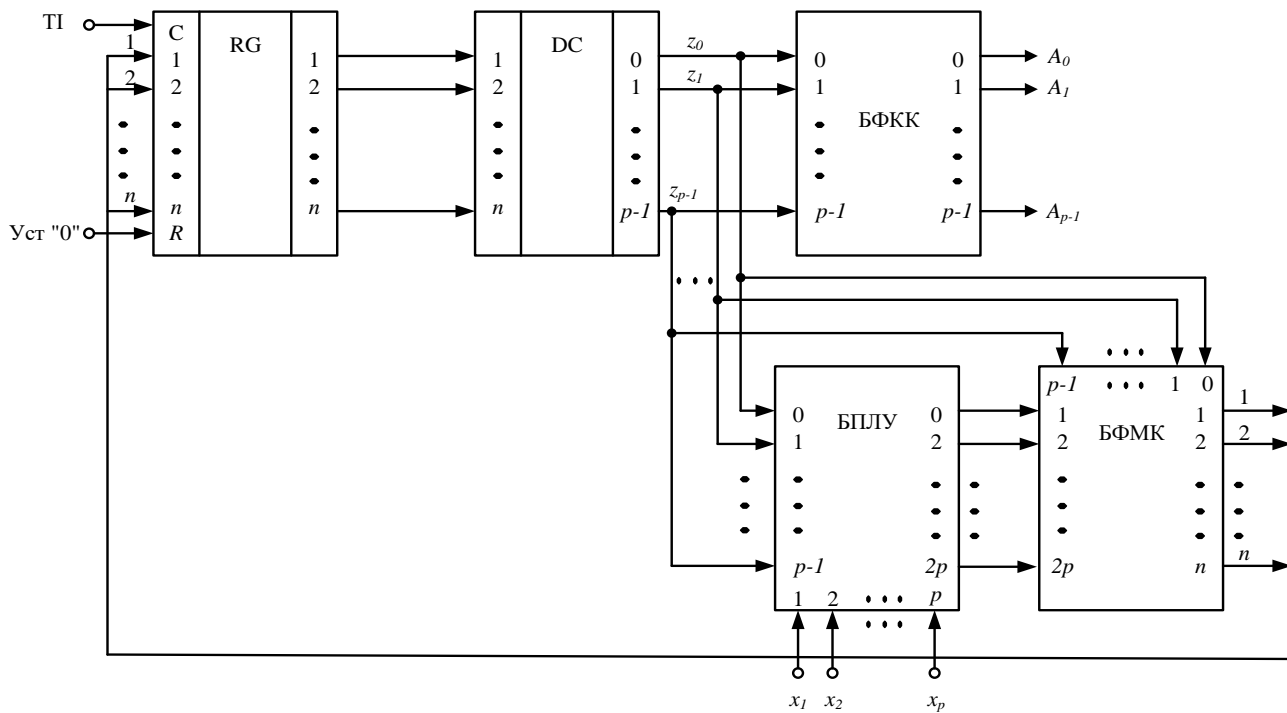


Рисунок 4.6 – Блок-схема мікропрограмного автомата Уїлкса

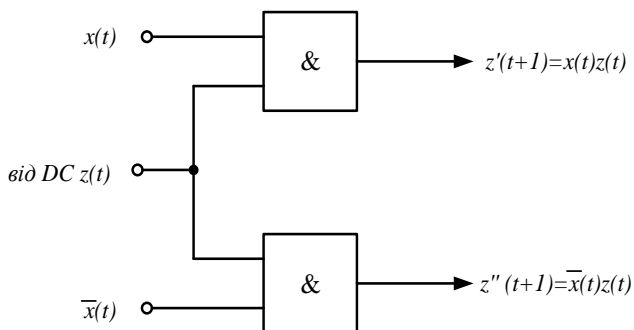


Рисунок 4.7 – Логічний вузол БПЛУ

На виходах схем I з'являється відповідно сигнал:

$$z'(t + 1) = x(t)z(t)$$

або

$$z''(t + 1) = \bar{x}(t)z(t),$$

який через БФМК задає мікрокоманду, яка буде виконана у наступний момент часу  $t+1$ . Потім цикл може бути повторений з іншим логічним вузлом, якщо немає повернення до попередньої мікрокоманди. Якщо після виконання будь-якої мікрокоманди не передбачається розгалуження у програмі, то відповідний логічний вузол буде відсутній і виконується наступна за порядком мікрокоманда, як це було показано у найпростішому мікропрограмному автоматі.

Якщо одна логічна умова пов'язана із другою, друга із третьою і т.д. то  $z'(t + 1)$  і  $z''(t + 1)$  будуть визначатися добутком декількох логічних змінних. Можливо також, що у мікропрограмі здійснюється звернення до однієї й тієї ж мікрокоманди при організації різних циклів. Це означає, що декілька логічних добутків повинні бути об'єднані за допомогою операцій логічного додавання.

Розглянемо побудову мікропрограмного автомата (МПА) Уїлкса на конкретному прикладі. Функціонування автомата задається за допомогою наступної ЛСА:



$$z_0 \ z_7 \downarrow^1 \ z_4 \ x_1 \uparrow^1 \ z_3 \ x_2 \uparrow^2 \ z_2 \ z_6 \downarrow^2 \ z_1 \ z_5 \ z_k.$$

На рисунку 4.8 наведена відповідна граф-схема алгоритму (ГСА) - із двома логічними умовами, що задаються змінними  $x_1$  і  $x_2$ , які моделюють датчики типу реле, тобто видають сигнали «ввімкнено» або «вимкнено», що відповідно кодуються 1 і 0. Також до ГСА входять мікрокоманди  $z_0, z_1, \dots, z_7$ , кодування яких наведено у таблиці 4.1.

Ставиться задача побудувати відповідний мікропрограмний автомат Уїлкса.

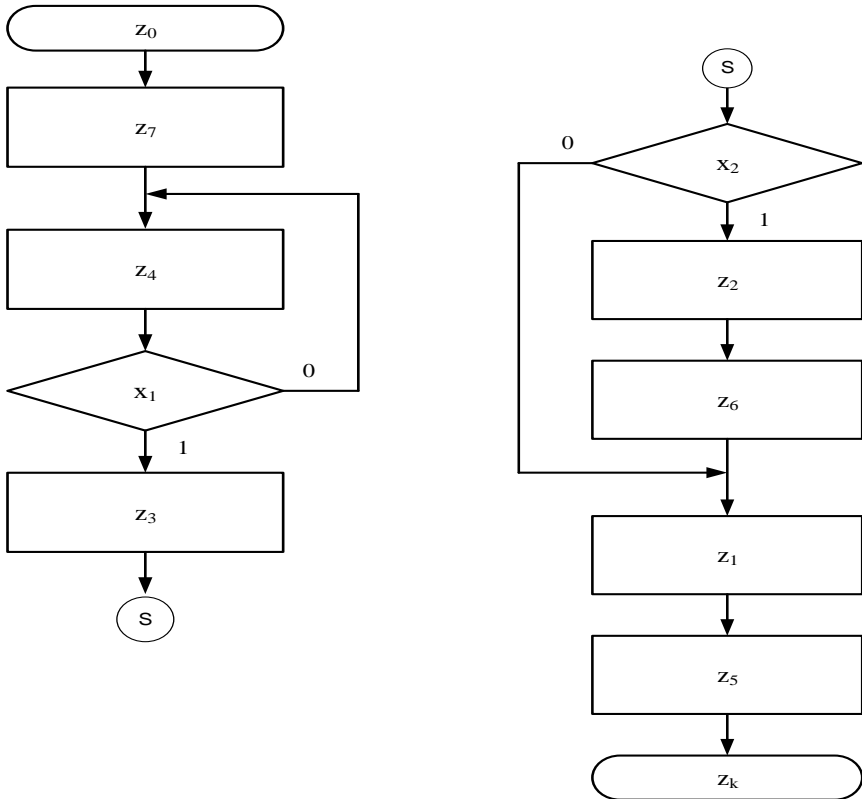


Рисунок 4.8 – Граф-схема алгоритму

Таблиця 4.1 – Кодування мікрокоманд

№ мікрокоманди	$M_k$	Код $M_k$
1	$z_0 = z_k$	0 0 0
2	$z_1$	0 0 1
3	$z_2$	0 1 0
4	$z_3$	0 1 1
5	$z_4$	1 0 0
6	$z_5$	1 0 1
7	$z_6$	1 1 0
8	$z_7$	1 1 1

Згідно ГСА побудуємо кодовану мікропрограму, яка формується наступним чином. Визначаємо першу мікрокоманду в момент часу  $t$  (табл. 4.2). Це буде мікрокоманда  $z_0$  (початок). Із ГСА бачимо, що після неї в момент часу  $t + 1$  іде мікрокоманда  $z_7$ . Це означає, що із нульової шини дешифратора, яка відповідає мікрокоманді  $z_0$ , на усі схеми АБО шифратора (БФМК) повинні бути заведені зв'язки (рис. 4.8). В результаті в момент часу  $t + 1$  на виходах АБО шифратора з'являється двійкова комбінація 111, яка надходить на входи  $D1, D2, D4$  регістра  $RG$  і за  $TI$ , що надходить на вхід  $C$  в момент часу  $t + 1$ , заноситься у регістр. Після мікрокоманди  $z_7$  аналогічно відпрацьовується мікрокоманда  $z_4$ . Після мікрокоманди  $z_4$  іде логічна умова  $x_1$ . У випадку, якщо вона дорівнює 1, то виконується команда  $z_3$ , у протилежному разі здійснюється повернення до команди  $z_4$  і цикл замикається. В автоматі на рис. 4.8 ця умова реалізується відповідним логічним вузлом, що складається із двох двовходових схем І. На їх об'єднаний вхід заведений зв'язок із четвертого виходу дешифратора  $z_4$ , а на інші входи подані логічні змінні  $x_1$  і  $\bar{x}_1$ . Як свідчить ГСА, вихід схеми, який відповідає добутку  $z_4 x_1$ , необхідно з'єднати із першою і другою схемами АБО шифратора БФМК, що приводить при виконанні логічної умови  $x_1 = 1$  до появи на його вході комбінації 011, яка відповідає мікрокоманді  $z_3$ . Для випадку  $x_1 = 0$  вихід другої

схеми I, на якому утворюється добуток  $z_4 \overline{x_1}$ , повинен бути з'єднаний із входом третьої схеми АБО. В результаті такого з'єднання на виході схем АБО утворюється комбінація 100, яка формує мікрокоманду  $z_4$ .

Аналогічно формується в автоматі решта зв'язків.

Подамо викладені результати за кодуванням мікропрограми у таблиці 4.2.

Таблиця 4.2 – Кодована мікропрограма

№ мікро-команди	$t$	$t+1$	Код $M_k$
1	$z_0$	$z_7$	1 1 1
2	$z_7$	$z_4$	1 0 0
3	$z_4 x_1$	$z_3$	0 1 1
4	$z_4 \overline{x_1}$	$z_4$	1 0 0
5	$z_3 x_2$	$z_2$	0 1 0
6	$z_3 \overline{x_2}$	$z_1$	0 0 1
7	$z_2$	$z_6$	1 1 0
8	$z_6$	$z_1$	0 0 1
9	$z_1$	$z_5$	1 0 1
10	$z_5$	$z_k = z_0$	0 0 0

На основі цієї таблиці складемо логічні рівняння для синтезу БПЛУ та шифратора БФМК:

- |    |                                      |     |                                      |
|----|--------------------------------------|-----|--------------------------------------|
| 1. | $z_7^{t+1} = z_0^t$                  | 6.  | $z_1^{t+1} = z_3^t \overline{x_2^t}$ |
| 2. | $z_4^{t+1} = z_7^t$                  | 7.  | $z_6^{t+1} = z_2^t$                  |
| 3. | $z_3^{t+1} = z_4^t x_1^t$            | 8.  | $z_1^{t+1} = z_6^t$                  |
| 4. | $z_4^{t+1} = z_4^t \overline{x_1^t}$ | 9.  | $z_5^{t+1} = z_1^t$                  |
| 5. | $z_2^{t+1} = z_3^t x_2^t$            | 10. | $z_0^{t+1} = z_5^t$                  |

Проведемо логічне об'єднання рівнянь, що мають в лівій частині однакові індекси. Це будуть за індексом чотири - друге і четверте і за індексом один - шосте і восьме рівняння:

1.  $z_7^{t+1} = z_0^t$
2.  $z_4^{t+1} = z_7^t \vee z_4^t \bar{x}_1^t$
3.  $z_3^{t+1} = z_4^t x_1^t$
4.  $z_2^{t+1} = z_3^t x_2^t$
5.  $z_1^{t+1} = z_3^t \bar{x}_2^t \vee z_6^t$
6.  $z_6^{t+1} = z_2^t$
7.  $z_5^{t+1} = z_1^t$
8.  $z_0^{t+1} = z_5^t$

Реалізуючи за цими рівняннями БПЛУ та шифратор БФМК, одержимо мікропрограмний автомат, який відповідає заданій ГСА (рис. 4.9).

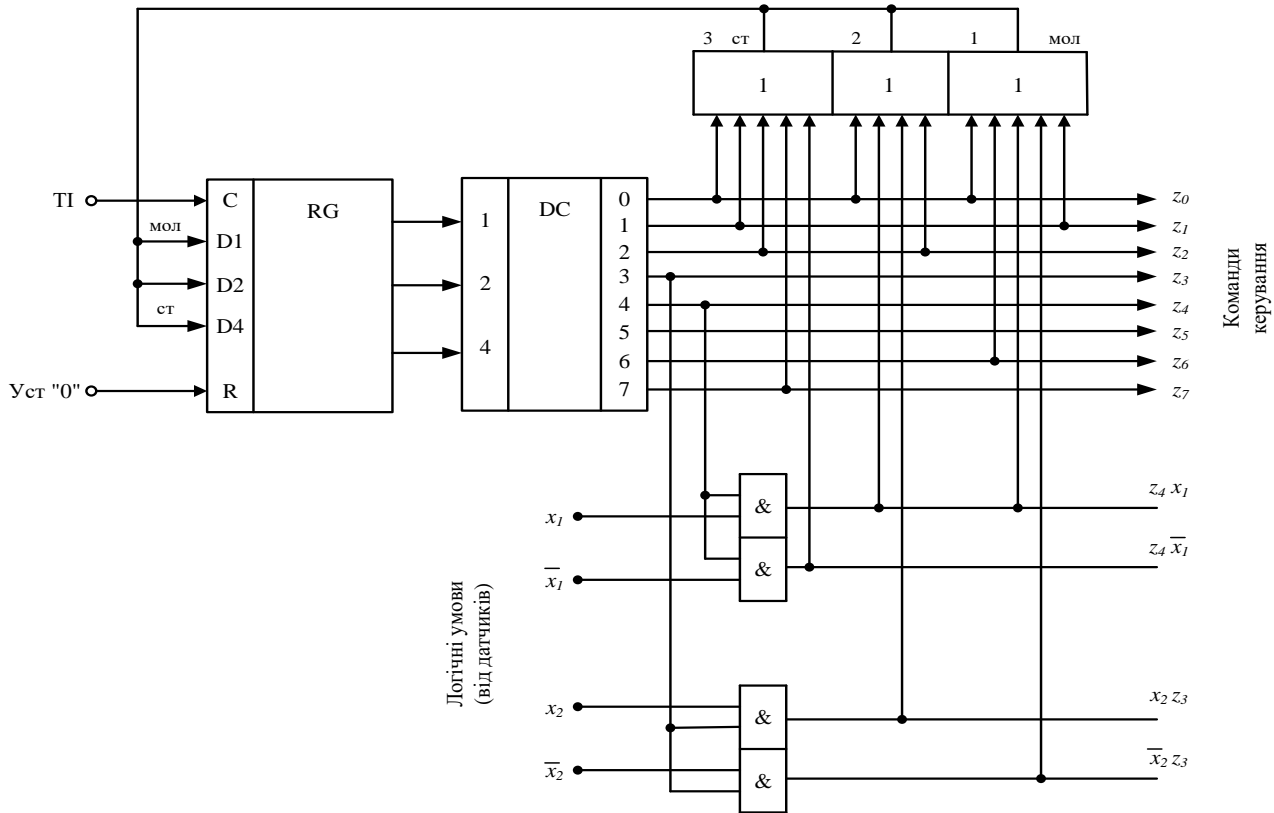


Рисунок 4.9 – Функціональна схема мікропрограмного автомата Уілкса

## СПИСОК ЛІТЕРАТУРИ

- 1 Борисенко О.А. Керуючі системи: Навчальний посібник. – Київ: Центр навчальної літератури, 2004 – 216 с.
- 2 Протасова Т.О. 5549 Методичні вказівки для лабораторних робіт із дисципліни «Керуючі системи» для студентів спеціальності 171 «Електроніка» усіх форм навчання / Т. О. Протасова, О. В. Д'яченко, О.А. Борисенко – Суми : Сумський державний університет, 2023. – 44 с.
- 3 Кравець П.І., Шимкович В.М., Бердник Ю.М. Інформаційно-керуючі системи. Лабораторний практикум: Навчальний посібник. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 142 с.
- 4 Єсаулов С. М. Аналіз, синтез і проектування цифрових систем керування : навч. посібник / С. М. Єсаулов, О. Ф. Бабічева; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 150 с.
- 5 Білінський, Й. Й. Цифрова схемотехніка. Електронно-обчислювальні пристрої: навчальний посібник / Й. Й. Білінський, Б. П. Книш. – Вінниця : ВНТУ, 2021. – 66 с.
- 6 Подчашинський Ю.О., Шавурський Ю.О., Лугових О.О. Проектування та конструювання пристроїв та систем управління: Навчальний посібник. – Житомир: ЖДТУ, 2018. – 280 с.
- 7 Бондаренко І.М., Бородін О.В., Карнаушенко В.П. Мікропроцесорні системи контролю та керування: Навч. посібник для студентів ЗВО. – Харків: ХНУРЕ. – 2020. – 244 с.
- 8 Борисенко О. А. Цифрова схемотехніка : підручник / О. А. Борисенко. – Суми : Сумський державний університет, 2016. – 200 с.
- 9 Борисенко О. А. Дискретна математика : підручник / О. А. Борисенко. – Суми : ВТД «Університетська книга», 2019. – 255 с.

Електронне навчальне видання

**Протасова** Тетяна Олександрівна,  
**Борисенко** Олексій Андрійович  
**Д'яченко** Олексій Вікторович

## **КЕРУЮЧІ СИСТЕМИ**

Конспект лекцій  
для студентів спеціальності  
171 «Електроніка»  
усіх форм навчання

Відповідальний за випуск А. С. Опанасюк  
Редактор С. М. Симоненко  
Комп'ютерне верстання Т.О. Протасова

Формат 60x84/16. Ум. друк. арк. . Обл.-вид. арк.

Видавець і виготовлювач  
Сумський державний університет,  
вул. Харківська, 2, м. Суми, 40007  
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.