Міністерство освіти і науки України
Сумський державний університет
Факультет електроніки
та інформаційних технологій

# КЕРУЮЧІ СИСТЕМИ

Конспект лекцій

для студентів спеціальності *171 «Електроніка»*
всіх 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. – 100 с.

Кафедра електроніки і комп’ютерної техніки факультету ЕлІТ

**ЗМІСТ**

С.

## Розділ І. Вступ до теорії керування

### Лекція 1. Основні поняття в керуванні

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

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

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

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

Наприклад, у хворого станом буде набір суб'єктивних та об'єктивних показників, таких, як самопочуття, температура, тиск, результати аналізу крові та ін. Стан військової операції може визначатися кількістю втрат у себе та 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 з несправного стану в справний, є керуючою дією. Також нею є звільнення й прийняття на роботу персоналу та робітників на виробництві, змінення номенклатури продукції, що випускається, накази командира тощо.

Керуюча дія передається через ***сигнал***, що надходить на вхід об'єкта керування та впливає на його вихідну величину. Характер її зміни за часом називають ***законом керування.***

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

Метою або завданням керування може бути, наприклад, одержання бажаного ефекту під час лікування хворого; досягнення високої якості продукції, що випускає підприємство, або зниження її собівартості; підтримання заданого ступеня

матеріального добробуту суспільства; здобуття переваги над противником у бою.

Саме керування, здійснюване за часом, є цілеспрямованим ***процесом*** із вироблення керуючих дій, спрямованих на змінення станів об'єкта керування, що ґрунтуються на опрацюванні інформації про нього.

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

Для процесу керування необхідно знати й передбачити поведінку об'єкта керування в разі різних можливих зовнішніх дій на нього.

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

Збурюючі фактори звичайно є випадковими величинами й 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 в разі появи 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, якщо альтернатив багато, то далеко не завжди серед них знайдеться така, яка збігається з метою керування. Тому воно можливе лише 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 називають його ***керованістю.***

Об'єкт вважають керованим, якщо існує керування *U(t)*, що 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 учений Р. Е. Калман. Він увів поняття ***повної*** та ***часткової*** керованості.

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

Отже, 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 наявних керівних дій.

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 й 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, що враховують не лише дію на об'єкт на даний момент часу, а й на попередні. Для побудови математичних моделей використовують диференційні рівняння, а також інші складні математичні підходи та методи.

Найбільш вивчені об'єкти керування, що мають технічну природу. Відповідно й 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 із них – принцип керування *за збуренням*, відомий також як *принцип розімкненого керування*, а також як *принцип Понселе – Чикалева*.

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

Рисунок 1.1 ілюструє принцип керування за збуренням.

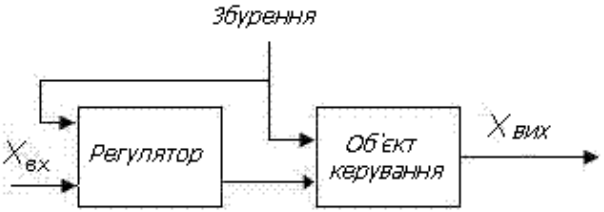



Рисунок 1.1 – Структурна схема принципу *розімкненого керування*

***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. Водночас припускається, що завчасно відомі ті чи інші збурюючі фактори. Цей принцип також використовують під час польоту на автопілоті, керування по копіру в ткацьких верстатах, під час пошиття 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 в резервуарі

тощо. Для забезпечення надійного керування ***потрібно враховувати кожну*** з величин збурення та для кожної будувати своє коло регулювання. Це практично здійснити неможливо, адже на роботу будь-якого об'єкта може впливати безліч причин. По-друге, для здійсненням керування за збуренням необхідно повністю знати залежність реакції системи на збурюючу дію будь-якої величини й увести цю залежність в алгоритм роботи регулятора. Тобто розроблення 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 в разі ретельного вимірювання збурень та врахування 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 межами. Математик знайде можливість займатися математикою в невідповідних для цього умовах.

Жан-Віктор Понселе (Jean-Victor Poncelet) народився в 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 Жана-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 в підготовці інженерних кадрів. Тут треба розуміти, що на початку XIX століття Франція була країною прикладчиків. 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 й потрапив до Саратова, де почав займатися математикою.

Понселе повернувся до Парижа, зробив ще багато великих відкриттів і навіть устиг 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 проводили «на папірці».

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

**Основні праці Ж.-В. Понселе:**

- *(1822) Traité des propriétés projectives des figures;*
- *(1826) Cours de mécanique appliqué aux machine;s*
- *(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*.

Наступний принцип керування є основою всієї сучасної теорії керування. Його називають ***принципом негативного зворотного зв'язку***, або просто ***принципом зворотного зв'язку***. Уперше застосували цей принцип керування незалежно один від одного у 1765 р. І. І. Ползунов для регулювання рівня води і в 1784 р.– Дж. Уатт для відцентрового 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 його називають ще принципом керування ***за відхиленням***.

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

Принцип зворотного зв'язку використовують тоді, коли вплив збурюючих факторів значний і вони раніше не відомі, так що знехтувати ними або їх урахувати практично неможливо. Керування здійснюють у цьому разі за кінцевим сумісним результатом впливу керуючих дій та збурюючих факторів. У цьому якраз і полягає цінність принципу зворотного зв'язку, для використання якого немає необхідності в попередньому вимірюванні збурюючих факторів і дослідженні характеру потрібних керуючих дій. У результаті їх вплив ураховується в процесі керування за знаком і величиною відхилення сигналу зворотного зв'язку. Цей сигнал дозволяє стежити за ефектом регулювання та усувати недо- та перерегулювання, збільшуючи тим самим точність керування. Водночас, якщо сигнал зворотного зв'язку збільшується, то керуюча дія повинна зменшитись і навпаки, якщо сигнал зворотного зв'язку зменшується, то керуюча дія збільшується. Це ї є проявом *негативного* зворотного зв'язку.

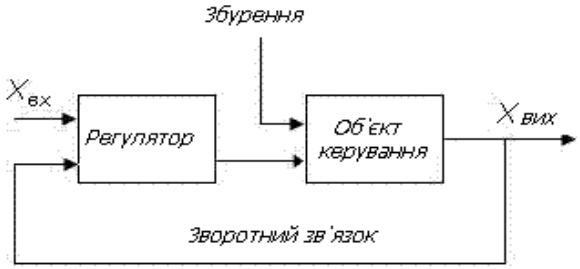



Рисунок 1.2 – Структурна схема принципу *негативного зворотного зв'язку*

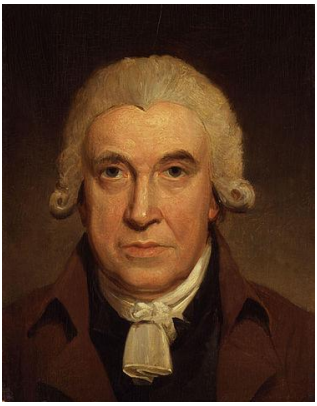
Прикладом дії принципу зворотного зв'язку є керування автомобілем. Водій неперервно аналізує свої керуючі дії та вилив збурюючих факторів за сигналами зворотного зв'язку, які надходять до нього як через прилади, так і через вікна автомобіля. Реагуючи на ці сигнали, він виробляє ті чи інші керуючі дії,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

Іншим прикладом керування зі зворотним зв'язком є керування бойовою операцією. Якщо залишити командира від результатів його наказів 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 та зазвичай неузгоджено.

Недоліком керування зі зворотним зв'язком є можливість утрати його стійкості. У цьому разі регулюючі дії не встигають за зміненнями величини, що регулюється, і тоді замість послаблення впливу керуючих сигналів на виході об'єкта керування за допомогою негативного зворотного зв'язку може здійснитися їх підсилення і як наслідок – трата стійкості керування.

Наприклад, в аварійній ситуації у разі 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***



***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 – у всьому 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 вимірювання. 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)*** використовують під час керування великою кількістю складних об’єктів. Керування кожним із них здійснюється здебільшого самостійно. Цей принцип застосовують для прискорення процесу ухвалення рішення. Водночас право ухвалення рішення за конкретними питаннями діяльності передається тому рівню керування, на якому ця діяльність здійснюється. Децентралізоване керування дозволяє покращити якість ухвалення рішень, ухвалених як на вищому, так і на нижніх рівнях. Водночас здійснюється робота з навчання та покращення професійних навичок штатного складу.

***Граф Гельмут Карл Бе́рнхард фон Мо́льтке, Мо́льтке Ста́рший*** (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 із Бісмарком і Рооном вважають 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 одержав усі можливі нагороди, усього 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 одержав 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 в 1800 році (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*** застосовують у разі, 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 керування організацією.

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

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

***До недоліків*** належать:

- придушення творчої ініціативи персоналу у вирішенні виробничих завдань організації;
- зниження оперативного керування;
- зниження можливості адаптації персоналу до нових умов виробництва й роботи.

Принцип ***екстремального керування*** застосовують у ситуації, 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 розроблення програмного забезпечення. Автор ***XP*** не придумав нічого нового, а взяв кращі практики гнучкої розробки й посилив до максимуму. Тому програмування й називають екстремальним.

Автор методики – американський розробник Кент Бек. У кінці 90-х років він керував проєктом *Chrysler Comprehensive Compensation System* і там уперше застосував практики екстремального програмування. Свій досвід та створену концепцію він описав у книзі *Extreme Programming Explained,* опубліковану в 1999 році. За нею були надруковані й інші книги, у яких докладно було описано практики ***XP.*** До становлення методології причетні також *Уорд Каннінгем, Мартін Фаулер* та інші.

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

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

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

Тому XP добре підходить для складних і невизначених проєктів. У традиційних методах розроблення вимоги до розвитку системи визначають на початку роботи над проєктом, і часто виправляють пізніше. Це означає, що вартість проєкту через зміни буде більшою за заплановану. XP використовують для скорочення вартості змін завдяки представленню простих значень, принципів і методів. Під час використання

екстремального програмування, проєкт повинен стати гнучкішим щодо змін.

Методологія XP будується навколо чотирьох процесів: кодування, тестування, дизайну та 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 й активно сприяти незалежно від рівня досвіду, а менеджери повинні бути зосередженими на зв'язках і відношеннях. У результаті ми одержуємо метод, який потенційно володіє високою адаптацією до серйозно та часто мінливих вимог до проєкту, але водночас не позбавлений ряду принципових недоліків і дуже високого ступеня залежності від людського фактора.

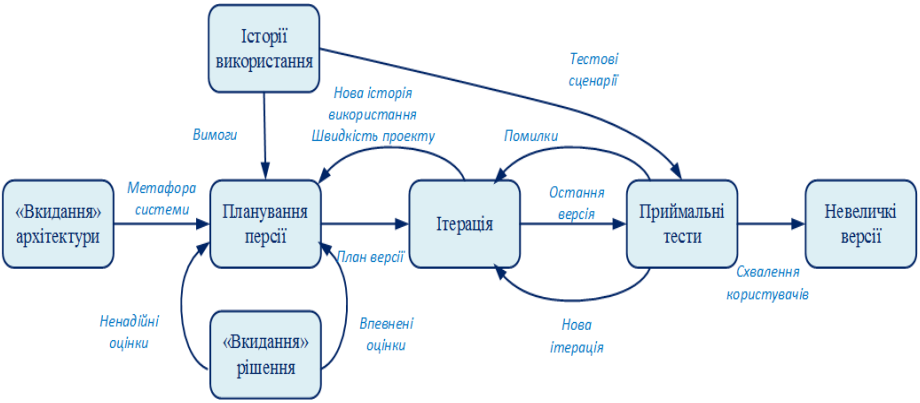



Рисунок 1.3 – Схема потоку робіт в XP

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 й доступний для питань.

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

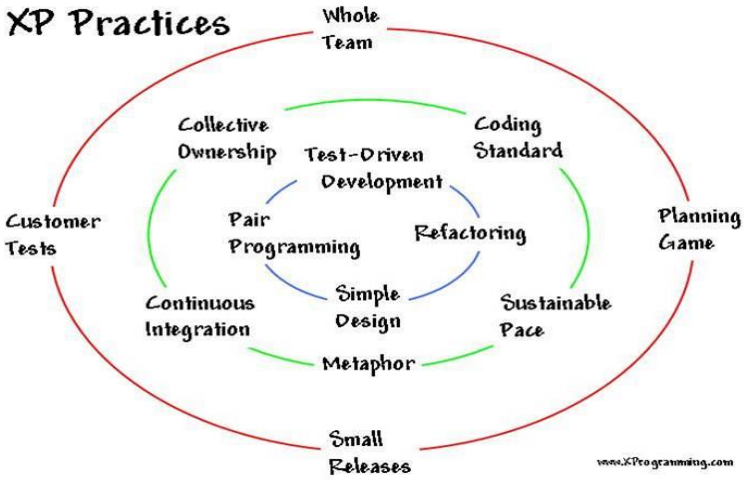



Рисунок 1.4 – Кругова схема-ілюстрація прийомів екстремального програмування

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 у процесі роботи над проєктом 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 розробникові 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. Водночас кожна версія повинна бути достатньо осмисленою з погляду корисності для бізнесу.

3) **Розуміння, що розділяється всіма**

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

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

Колективне володіння кодом (Collective code ownership) або обраними шаблонами проєктування (Collective patterns ownership)-означає, що кожен член команди несе відповідальність за весь вихідний код. Отже, кожен має право вносити зміни в будь-яку ділянку програми. Парне

програмування підтримує цю практику: працюючи в різних парах, усі програмісти знайомляться з усіма частинами коду системи. Важлива перевага колективного володіння кодом у тому, що воно прискорює процес розроблення, оскільки під час появи помилки її може усунути будь-який програміст.

Стандарт кодування (Coding standard or Coding conventions) – у межах XP необхідно домогтися того, щоб було складно зрозуміти, хто є автором тієї чи іншої ділянки коду, уся команда працює уніфіковано, як одна людина. Команда повинна сформувати набір правил, а потім кожен член команди повинен слідувати цим правилам у 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).

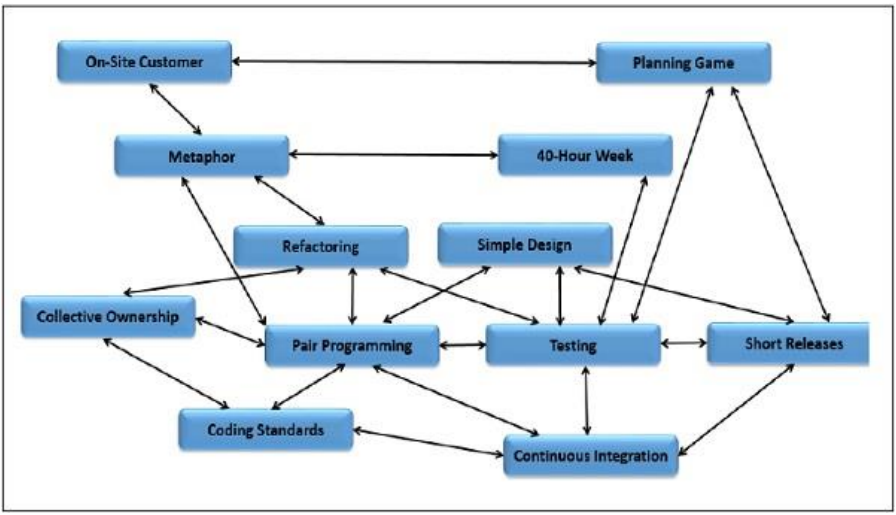



Рисунок 1.5 – Стрілкова діаграма-ілюстрація прийомів екстремального програмування

***Переваги екстремального програмування:***

- замовник одержує саме той продукт, який йому потрібен, навіть якщо на початку розроблення сам точно не уявляє його кінцевого вигляду;
- команда швидко вносить зміни в код і додає нову функціональність за рахунок простого дизайну коду, частого планування й релізів;
- код завжди працює за рахунок постійного тестування та безперервної інтеграції;
- команда легко підтримує код, тому що він написаний за єдиним стандартом;
- швидкий темп розроблення за рахунок парного програмування;
- висока якість коду;
- знижуються ризики, пов'язані з розробленням, тому що відповідальність за проєкт розподілена рівномірно.

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

***Недоліки екстремального програмування:***

успіх проєкту залежить від залучення замовника, якого не так просто домогтися, важко передбачити витрати часу на проєкт, тому що на початку ніхто не знає повного списку вимог;

- успіх XP сильно залежить від рівня програмістів, методологія працює лише із senior фахівцями;
- менеджмент негативно ставиться до парного програмування, не розуміючи, чому він повинен оплачувати двох програмістів замість одного;
- регулярні зустрічі з програмістами дорого обходяться замовникам;
- вимагає занадто значних культурних змін;
- через нестачу структури й документації не підходить для великих проєктів.

***Цікавий матеріал про екстремальне програмування:***
*http://www.info-system.ru/article/article_extrime_prog.html*
*https://ppt-online.org/665110*
*https://worksection.com/blog/extreme-progmamming.html*
*https://worksection.com/ua/blog/extreme-programming.html*
*https://wiki.cuspu.edu.ua/index.php(XP)*
*https://elearning.sumdu.edu.ua/free_content/lectured*

Принцип ***адаптивного керування*** використовують у разі, коли процес керування необхідно 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 залежно від змін параметрів об'єкта керування або збурюючих факторів, що впливають на об'єкт керування. Системи, побудовані за таким принципом, називають адаптивними. Адаптивне керування досить поширене в багатьох застосуваннях теорії керування.

***За характером змін керуючі пристрої*** адаптивні системи поділяють 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 із сигнальною (пасивною);
- системи з активною (параметричною) адаптацією;

- комбіновані системи (якщо система об’єднує в собі обидва види адаптації).

Принцип ***випадкового керування*** виникає в неповністю визначених ситуаціях. Керування вибирається з деяких можливих стратегій здебільшого випадково.

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

Принцип ***програмного керування*** використовують у разі, коли необхідно, щоб стани об’єкта керування змінювались згідно із *заздалегідь* заданим законом. У цьому випадку між станами об’єкта керування та керівними впливами встановлюється функціональний зв’язок. Закон зміни керівних впливів встановлюється при цьому *апріорно*.

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

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

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

Розглянуті принципи керування вирішують за допомогою *систем керування*.

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

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

Так, наприклад, водій та автомобіль сумісно утворюють систему керування, у 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 тощо.

Керуюча система та об'єкт керування з'єднуються через їх виходи та 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 загалом із керуючих засобів, елементів порівняння, інформаційної системи, системи збирання та передавання даних, виконавчих та регулювальних 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 та 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 таких 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, зокрема й ЕЦОМ, які виробляють дії на об'єкт керування за певним законом.

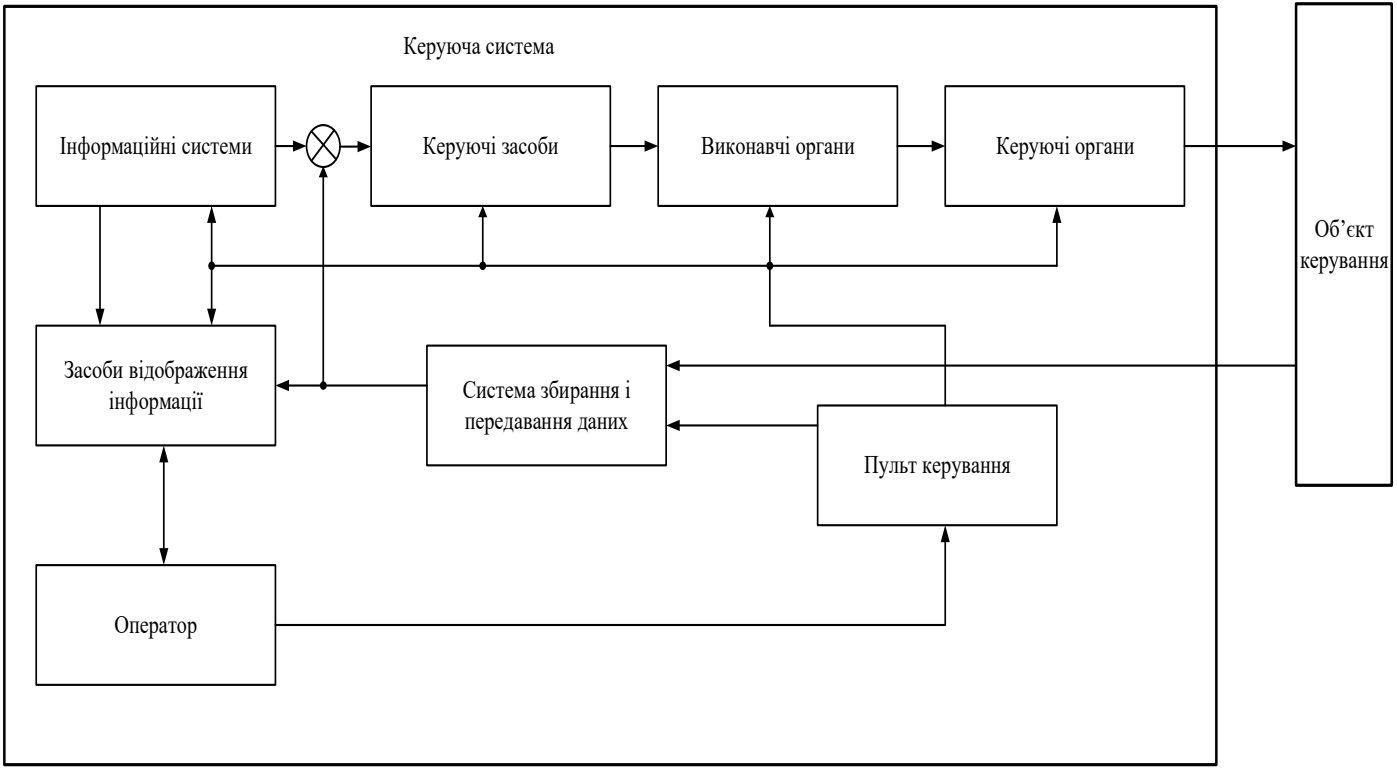



Рисунок 1.6 – Структура системи керування

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

Регулювальні органи являють собою різні засувки, заслінки, клапани, помпи тощо. Вони змінюють, наприклад, витрату палива, температуру, тиск пари, величину струму, навантаження 1 кількість обертів двигуна.

Наприклад, в автомобілі регулюючими органами будуть кермо, педаль газу, гальма, а виконавчими – руки й 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, Для підвищення ефективності ухвалення 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 впродовж 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, а також здібність до самонавчання. Тому багато систем керування не можуть обійтися без участі в них людини.

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

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

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

Так, в автомобілі системи збирання й передавання даних розв'язують задачі інформування водія про кількість бензину у баку, швидкість руху, температуру двигуна тощо.

На елементі порівняння одержані сигнали порівнюються з тими, що надходять від інформаційної системи. У результаті визначається величина розходження – відхилення реальних параметрів об'єкта стосовно необхідних їх значень.

За допомогою керуючих засобів це розходження усувають шляхом вироблення ними відповідних дій на виконавчі пристрої, які зі свого боку діють на регулювальні органи.

Останні змінюють параметри об'єкта керування так, щоб існуюче розходження знизити до мінімуму, а в ідеалі – до нуля, у чому й полягають мета й завдання будь-якого керування.

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

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

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

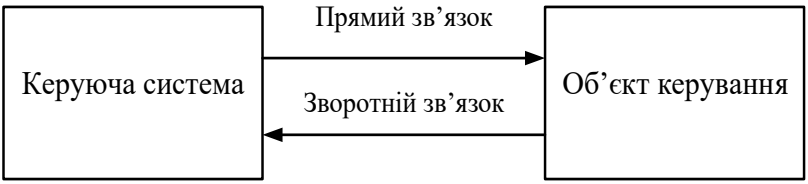



Рисунок 1.7 – Узагальнена структура системи керування

Зв’язок від керуючої системи до об’єкта керування називають *прямим*, а від об’єкта керування до керуючої системи – *зворотним*.

За допомогою прямого зв’язку здійснюється процес керування об’єктом – зміна 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, а отже, визначає подальші дії керуючої системи щодо зміни станів об’єкта керування.

Зазвичай процес керування здійснюється так: від об’єкта керування до керуючої системи передається інформація про значення параметрів об’єкта керування. Керуюча система порівнює їх із потрібними значеннями параметрів, що зберігаються в її пам’яті, і визначає величину їх розходження. Потім вибирають 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, ураховуючи 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 під час проєктування системи керування, є підвищення її *безпеки*, *надійності*, *завадостійкості* та *живучості*.

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

Здібність системи керування виконувати свої функції впродовж заданого відрізка часу, називають її *надійністю*.

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

Можливість виконання системою керування своїх основних функцій у разі відмови частини обладнання називають *живучістю*.

Сучасні системи керування так чи інакше вирішують ці завдання, однак вимоги до них неперервно зростають, і тому боротьба із зовнішніми та 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, у межах якої можна дослідити особливості тих чи інших їх структур. В основу класифікації систем керування покладені різні ознаки, наприклад, види сигналів, принципи й типи керування. Розглянемо більш поширені класифікаційні ознаки й відповідні класи систем керування.

Насамперед системи керування поділяють за ознакою типу об'єкта керування, яким вони керують.

Основі об'єкти керування можна поділити на *біологічні, технічні* та *соціально-економічні*. Відповідно й системи керування поділяють 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 й виробництва, а також різні пристрої та обладнання.

Третій клас становлять соціально-економічні утворення – установи, армія, держава тощо.

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

За формою сигналів, що використовуються системами керування, їх поділяють на *аналогові* та *цифрові*.

У недавньому минулому найбільш поширеними були аналогові системи керування. Їх особливістю є використання аналогових величин як носіїв інформації: напруги, струму,

частоти, фази тощо. Перевагою аналогових систем керування є неперевершена швидкодія та порівняно недорога й компактна їх реалізація. Однак точність і надійність роботи цих систем були недостатніми для все більш нових їх застосувань на практиці, тому порівняно недавно були розроблені цифрові системи керування, основою роботи яких стало цифрове оброблення сигналів і відповідно цифрове їх подавання. Їх головна перевага – це висока точність роботи. разом із цим значно зросли їх надійність, завадостійкість і живучість. Крім того, цифрові системи керування мають небачену для аналогових систем гнучкість га універсальність. Швидкодія цифрових систем керування хоча й менша від швидкодії аналогових, але для більшості практичних випадків виявляється достатньою. У 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, у багатьох сферах практичного застосування, де вимоги до точності не настільки високі й необхідна висока швидкодія або мала вартість, достатньо ефективно використовують аналогові системи керування.

За видом зв'язку системи керування поділяють на системи з *прямим* і *зворотним* зв'язком. У першому випадку інформація передається лише 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 називають ще системами з *розімкненим* зв'язком, а зі зворотним – із *замкненим*.

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

У більш складних, найбільш поширених випадках, використовують системи керування зі зворотним зв'язком, тому

що вони не потребують повної інформації про збурюючі дії та всі характеристики об'єкта керування.

Однак наявність зворотного зв'язку може призвести до зниження, а то й до втрати стійкості системи керування У 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* називають *автоматизацією*.

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

Під час 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 разі автоматизації всіх агрегатів та устаткувань, що беруть участь у робочому процесі.

Відповідно до видів автоматизації системи керування поділяються на системи з *частковою*, *комплексною* та *повною* автоматизацією.

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

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

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

Автоматизовані системи керування залежно від об'єкта керування поділяють на ряд підкласів. Із них найбільш відомі *автоматизовані системи керування підприємством* (АСКП) та *автоматизовані системи керування технологічним процесом* (АСКТП).

АСКП є системами керування виробничо-господарчою діяльністю підприємства, що ґрунтується на комплексному використанні економіко-математичних методів та сучасних засобів оброблення інформації. Необхідність створення та втілення АСКП пов'язана з великою кількістю об'єктів керування, масштабністю виробництва й 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 разі, коли через будь-які причини неможливо автоматизувати всі задачі керування, і тоді для їх розв'язування звертаються до людини. Вона звичайно приймає остаточне рішення, а завчасне оброблення інформації та її збирання здійснюють зазвичай цифрові пристрої та машини.

В АСКТП більшість контурів регулювання будують за ієрархічним принципом.

Перший нижній рівень ієрархії – це основні регулятори, що стабілізують технологічні параметри або змінюють їх відповідно до керуючих сигналів. Основні регулятори зазвичай безпосередньо впливають на виконавчі органи.

Другий рівень утворюють коректувальні 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 та змінюють завдання регуляторам, що знаходяться на нижніх рівнях ієрархії.

У разі повної або майже повної автоматизації використовують системи *автоматичного* керування (САК).

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

Метою автоматичного керування зазвичай є підтримка заданих значень керованих (регульованих) величин за умов повної автоматизації. Цієї мети досягають за допомогою об'єднаних у систему автоматичних пристроїв, що працюють без втручання людини. Вони вирішують більш прості завдання, ніж автоматизовані системи, однак із більшою швидкодією та точністю. Ці системи зазвичай є складовою частиною в автоматизованій системі керування, звільняючи людину від рутинної роботи і дають їй час для прийняття відповідальних рішень.

Так, наприклад, у сучасних автомобілях багато операцій, виконуваних раніше людиною, передаються автоматам. 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 й технологічними процесами в автоматичному режимі роботи. Вони входять як складова і важлива частина до керуючих систем і забезпечують виконання їх алгоритмів керування. Крім керуючих автоматів, до системи входять датчики, електроавтоматика, засоби зв'язку, інформаційна система, виконавчі та регулювальні органи. Разом з основною функцією керуючі автомати вирішують також завдання забезпечення такого режиму системи керування, у якому кожну команду можна здійснити за допомогою органів ручного керування. Керуючі автомати також здійснюють типові захисти, такі, як захист від самозапуску під час зникнення та появи знову живлення від електромережі, захист від перенавантажень, захист у разі несправностей або неправильних дій оператора, контроль справності пристрою, локалізацію та усунення несправностей, сигналізацію про хід роботи обладнання та в особливих випадках видачу сигналів у систему реєстрації, а також обліку роботи обладнання.

Керуючі автомати, крім шафи керування, звичайно, містять одну або декілька шаф електроавтоматики, що з'єднуються з обладнанням, що обслуговується, кабелями або джгутами проводів. Шафи керування та електроавтоматики виготовляють і монтують зазвичай 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 та одночасно із цим необхідно знати алгоритми керування, які уточнюють на всіх етапах розроблення й дуже часто змінюються на етапі налагодження системи, а інколи навіть і в процесі експлуатації.

Керуючі автомати можуть бути електричними, пневматичними, гідравлічними, електро- і пневмогідравлічними.

Крім того, вони підрозділяються на системи з *логічним* (релейним) та *аналоговим* принципами керування. Останні у своїй роботі разом із цифровим обробленням використовують також аналогове оброблення інформації. Їх застосовують у системах керування на підприємствах із неперервним циклом виробництва і є здебільшого складнішими, ніж перші.

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

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

У результаті виникла можливість реалізації в одній мікросхемі повної схеми того чи іншого керуючого автомата. Такі мікросхеми називали *замовленими*, однак при цьому виникло протиріччя між економічною доцільністю випуску таких схем великими серіями і здебільшого невеликою їх кількістю під час вирішення того чи іншого конкретного практичного завдання керування.

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

У результаті були розроблені *програмовані* керуючі автомати. Порівняно 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 із малим ступенем інтеграції. Тому хоча програмовані керуючі автомати й потіснили автомати із жорсткою логікою, однак у підсумку витіснити їх не змогли. Більше того, на сьогодні з появою нових інтегральних технологій вартість і час виготовлення замовлених мікросхем настільки знизилися, що в ряді випадків їх використання є економічно більш вигідним для завдань керування, ніж універсальних. Тому часто під час виконання відповідальних замовлень іде зворотний процес – заміна в ряді випадків програмованих керуючих автоматів автоматами на замовлених великих інтегральних схемах.

Отже, сучасні керуючі автомати поділяють на два основних класи – із *програмованою* (*гнучкою*) та зі *схемною* (*жорсткою*) реалізацією алгоритмів керування.

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

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

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

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

Водночас для вирішення більш простих завдань керування їх широко застосовують через низьку вартість, високу швидкодію та надійність.

Однак, існує ще один важливий клас керуючих автоматів, проміжний між програмованими автоматами й автоматами зі схемною логікою, які використовують мікропрограмний принцип керування – мікропрограмні керуючі автомати. Цей принцип уперше був використаний під час побудов ЕОМ для підвищення регулярності їх структури.

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

Мікропрограмний автомат є програмованим автоматом, у якому на відміну від звичайної або керуючої ЕОМ програмування ведеться на найнижчому 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 всі вхідні й 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 й програмістом, і схемотехніком.

Отже, мікропрограмування – це є деяка методика з реалізації функцій керування, в основу якої покладене впорядковане 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 у разі розроблення пристроїв керування;

3) простота процедури діагностики, виявлення та усунення несправностей;

4) простота проєктування, налагодження та експлуатації;

5) великий термін служби, що забезпечується за рахунок перепрограмування мікропрограмних пристроїв керування в в процесі їх експлуатації.

Водночас мікропрограмні автомати потребують більше часу для своєї роботи, ніж автомати зі схемною логікою, оскільки потрібен додатковий час для послідовного виконання операцій, як це здійснюється в ЕОМ.

Мікропрограмні керуючі автомати поєднують у собі переваги універсальних програмованих керуючих автоматів та автоматів зі схемною логікою. Вони, з одного боку, мають необхідну гнучкість та універсальність, а з іншого – потребують для своєї реалізації порівняно невеликої кількості апаратурних витрат і відрізняються високою швидкодією й 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 використовують під час програмування мовами високого рівня, широко застосовуваних у сучасних програмованих *мікропроцесорних* системах керування.

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

Отже, сучасні керуючі автомати можна поділити на автомати зі схемною та 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 та в поєднанні з мікропроцесорною технікою найбільш ефективними для вирішення складних завдань керування.

## Розділ ІІ. Подання алгоритмів роботи керуючих автоматів

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

Під час опису функціонування різних засобів обчислювальної техніки досить часто використовують їх подання у вигляді сукупності керуючого та операційного автоматів (див. рис. 2.1).

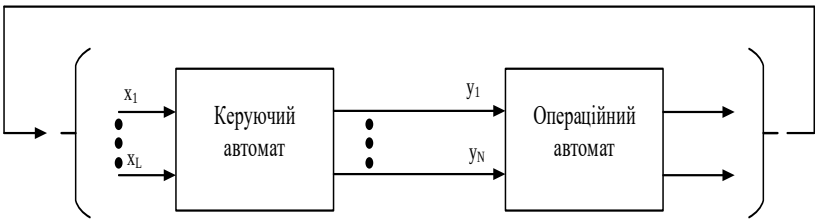



Рисунок 2.1 – Структура керуючого автомата

Так, в ЕОМ до операційного автомату потрібно віднести блоки пам'яті, регістри, суматори, канали передавання інформації, шифратори й таке інше, тобто всі пристрої, що виконують деякі операції, а до керуючого автомата – ту частину ЕОМ, яка координує дії перерахованих пристроїв, визначаючи послідовність оброблення в них інформації.

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

***Означення:*** графічне подання роботи автомата, у якому кожний із його етапів зображений геометричною фігурою, з'єднаних стрілками, що показують послідовність виконання

етапів, за необхідності з поясненнями, називають *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 лише одна. До кінцевого оператора $A_k$ можуть входити одна або декілька дуг, але не виходить жодна з них.

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

Вершини керуючих операторів відображають за допомогою прямокутників, усередині яких прописують словами керуючу команду або позначають символьно великими літерами латинського алфавіту, наприклад, $A_1$, $A_2$,…, $A_j$.

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

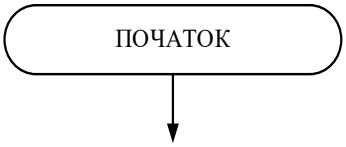
Керуюча вершина може мати лише один вихід, водночас логічна вершина має два виходи, які зазвичай позначають 0 і 1. Інколи їх заміняють словами «Ні» і «Так» відповідно. Це випливає з того, що результатом перевірки деякої логічної умови $x$ можливі два виходи: $x = 1$ або $x = 0$.

Зображення вершин ГСА наведені на рисунку 2.2.

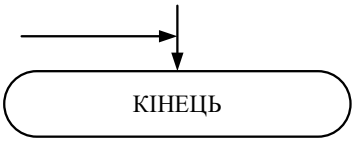
***Умови, які повинна задовольняти ГСА***

1. Мати одну початкову та одну кінцеву вершину, які не нумеруються.

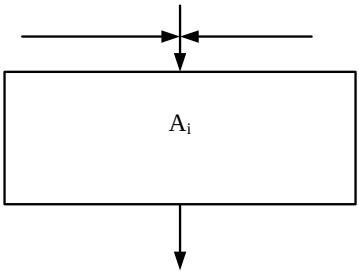
2. Містити кінцеву кількість вершин, кожна з яких є або початковою, або кінцевою, або керуючою, або логічною.



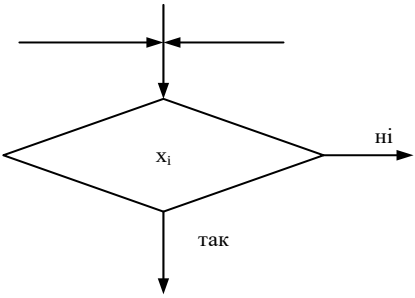

Початкова вершина



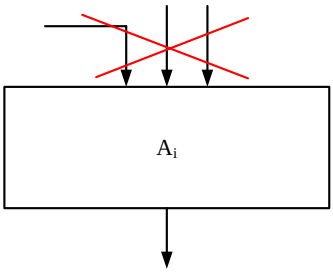

Кінцева вершина




Операторна (керуюча) вершина




Логічна (умовна) вершина



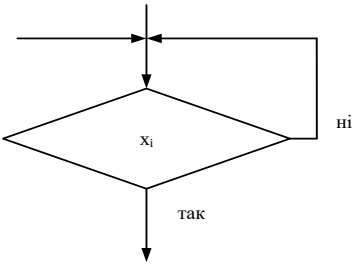




Рисунок 2.2 – Зображення вершин ГСА

3. Входи та виходи вершин з’єднуються між собою за допомогою дуг, спрямованих завжди від виходу до входу.

4. Кожний вихід об’єднаний лише 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 із виходів умовної вершини може бути з’єднаним із її входом, що неприпустимо для керуючої вершини.

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

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

Запис у вершині може бути символьним або змістовним.

Зазвичай під час проєктування пристроїв попередньо складають змістовну ГСА, у якій у 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 зображують початкову вершину ГСА. Після неї одержують першу операторну вершину, яка зазвичай є оператором уведення початкових даних. Потім викреслюються наступні вершини, які є як керуючими, так і логічними операторами, і всі зв’язки між ними. Останньою відображають кінцеву вершину.

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

За необхідністю здійснюють перехід до символьної 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 символьна ГСА для змістовної ГСА з рисунка 2.3.

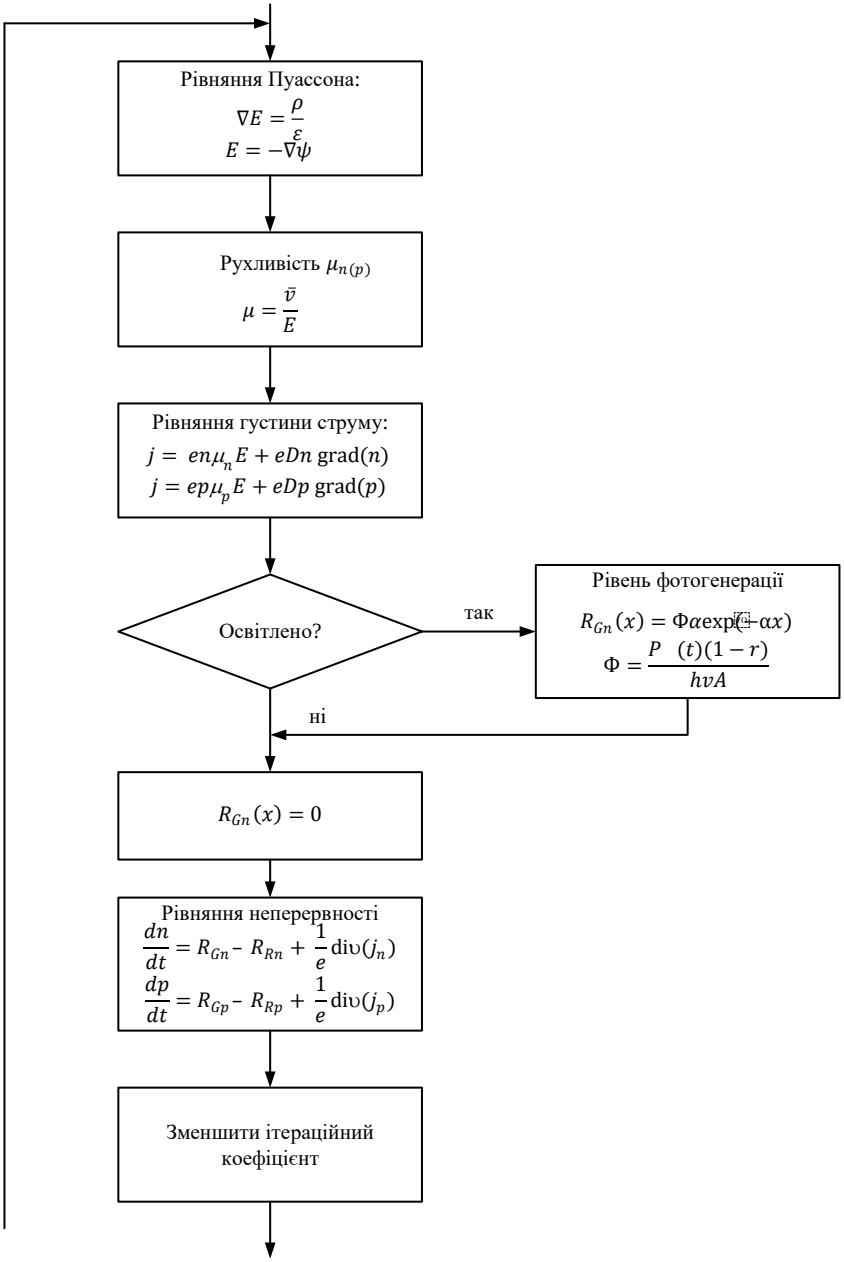



Рисунок 2.3 – Фрагмент змістовної ГСА

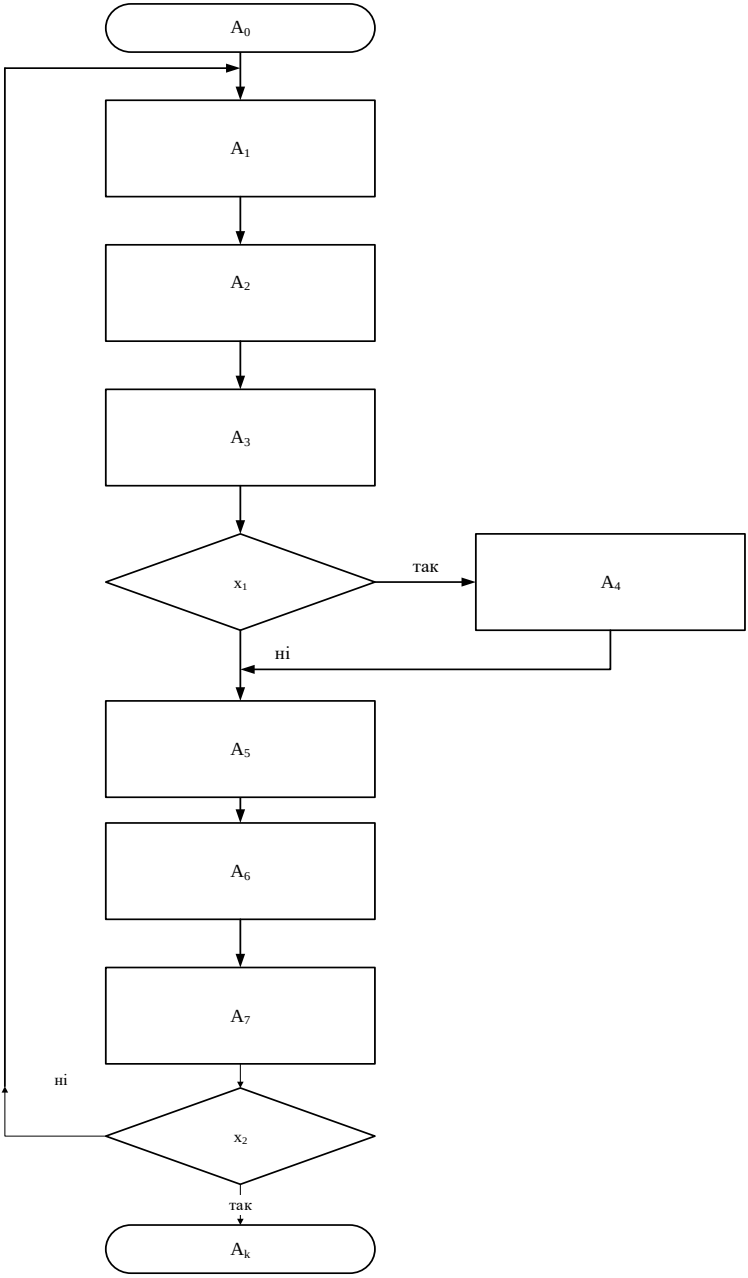



Рисунок 2.4 – Символьна ГСА

***Класифікація ГСА***

***Означення.*** ГСА, у якій відсутні логічні оператори, називають *простою* або *лінійною.*

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

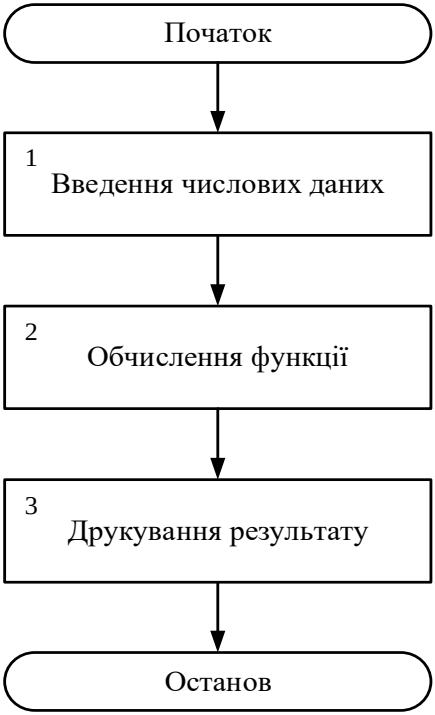



Рисунок 2.5 – Лінійна ГСА

***Означення.*** ГСА, у якій є логічні вершини, називають *розгалуженою.*

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

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

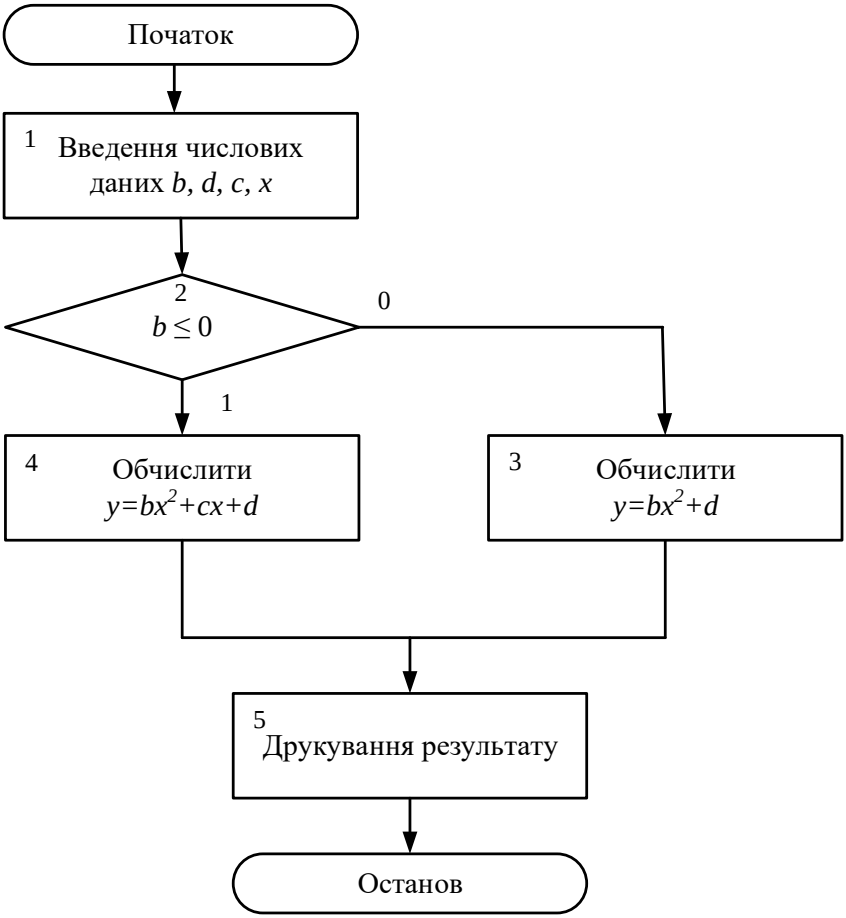



Рисунок 2.6 – Розгалужена ГСА

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

На рисунку 2.7 наведено приклад циклічної ГСА для функції

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

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

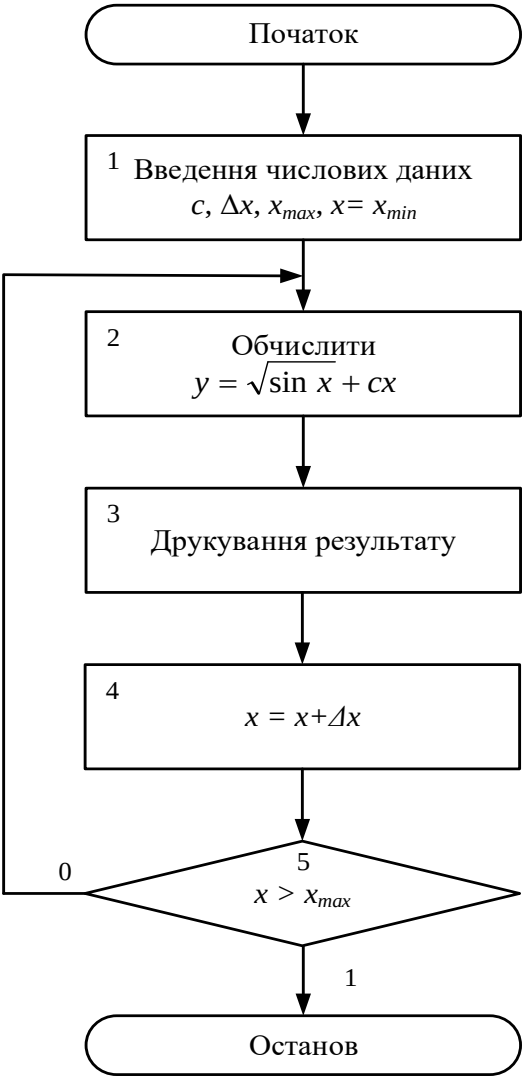



Рисунок 2.7 – Циклічна ГСА для обрахунку функції з даними, що змінюються

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

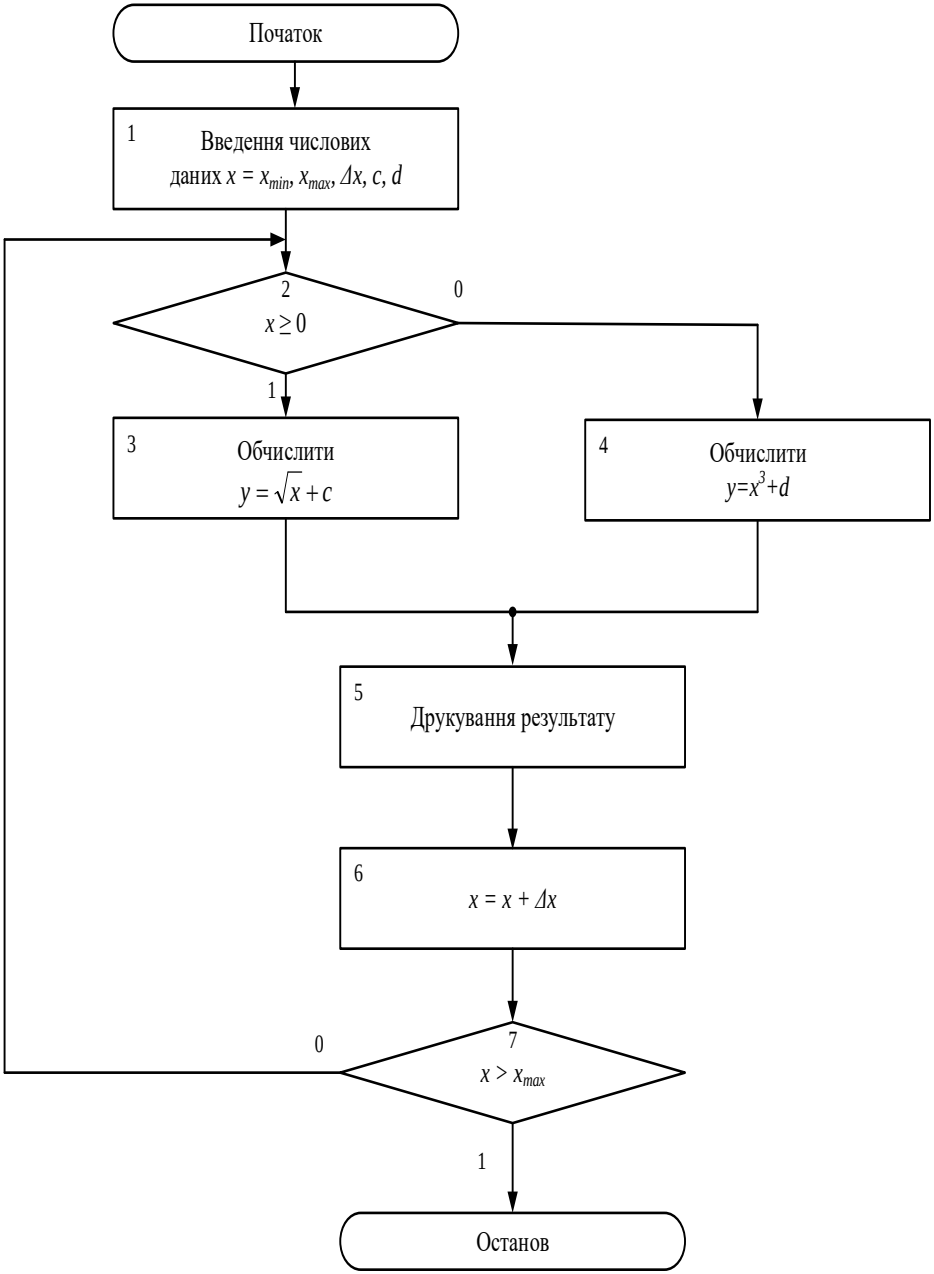



Рисунок 2.8 – Комбінована ГСА

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

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

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

***Завдання із зірочкою)))***

- Побудувати змістовну та символьну ГСА керуючого автомата 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 у 5 копійок кожна. Будемо вважати, що як операційні автомати використовують:

- 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 символьну 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 й зовсім непридатна для введення в ЕОМ. Тому часто після подання керуючого алгоритму у вигляді ГСА потрібне його перетворення на ЛСА.

***Означення.*** Вираз, складений із керуючих та логічних операторів, записаних один за одним, а також певним чином розставлених пронумерованих стрілок, називають *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*,... або однією й тією самою великою літерою, але з різними індексами, наприклад, $A_1, A_2, A_3, \ldots$ 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 $x_1, x_2, \ldots, x_n$ і можуть набувати лише двох значень 1 або 0.

Оскільки кожна логічна умова може набувати лише одного з двох значень 1 або 0, то потрібен символ, який би зазначав подальший порядок виконання операцій. Тому після кожної логічної умови стоїть стрілка, яка має вгорі номер. Вона 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 разі, якщо логічна умова $x_i = 0$. Кінець стрілки, що має також над собою справа номер, показує оператор, якому повинно бути в цій ситуації 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.

***Умови, які повинна задовольняти ЛС***

1. Містить один початковий та один кінцевий оператор, які не мають перед собою та 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 й 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 варіанти:

1) логічна умова має одиничне значення. У цьому варіанті 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 з операторів, що виконується, містить вказівку про зупинення роботи алгоритму або коли на деякому етап не виявляється оператор ЛСА, який повинен був би виконуватися.

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

## Лекція 8. Взаємні перетворення між граф-схемами й 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 й непридатна для введення в ЕОМ. Водночас логічна схема алгоритму (ЛСА) подається компактно у вигляді рядка, який складається з операторів, що послідовно виконуються, і тому зручна для кодування й запису в ЕОМ. Тому часто після подання керуючого алгоритму у вигляді ГСА потрібно його перетворення на ЛСА.

***Алгоритм перетворення ГСА на ЛСА містить такі кроки:***

• аналізують ГСА, для якої необхідно записати ЛСА. Вибирають шлях максимальної довжини;

• першим лівим оператором у рядку ЛСА записують початковий оператор;

• праворуч від написаного оператора записують наступний в ГСА за ним керуючий оператор або логічну умову;

• якщо в ГСА оператори, що виконуються, слідують один за одним, то відповідні оператори аналогічно один за одним записують у 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 зазначають номер. Загалом номер стрілки не збігається з індексом логічної умови. Праворуч від стрілки записують оператор, управління до якого передається в разі рівності одиниці логічної умови. Передача управління в разі рівності логічної умови нулю здійснюють за стрілкою. Кінець стрілки з відповідним номером проставляють перед оператором, якому передається керування;

• описана процедура повторюється до тих пір, поки в рядку ЛСА не буде записаний кінцевий оператор $A_к$;

• під час аналізу більш складних ГСА іноді виникають ситуації, коли жоден із можливих шляхів не проходить через усі вершини ГСА. Формування вставок в ЛСА для операторних вершин, що випадають під час аналізу, здійснюється за допомогою символу безумовного переходу $\omega$, який виконує роль знаків пунктуації в ЛСА.

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

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

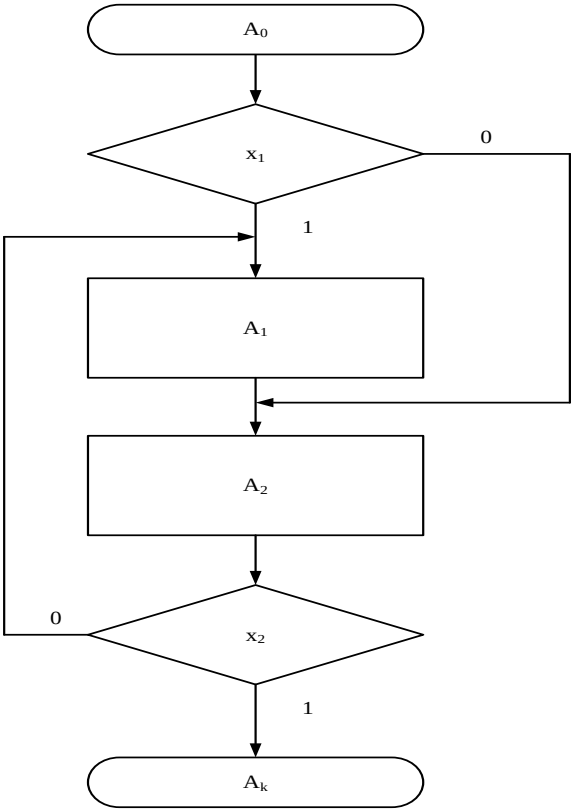



Рисунок 2.9 – Граф-схема алгоритму

***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 зазначити, що номер стрілки не має особливого значення й вибирається довільно. Єдине, що потрібно, щоб не було двох стрілок з однаковими номерами. Справа після стрілки проставляють 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 разі 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 із зірочкою)))***

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

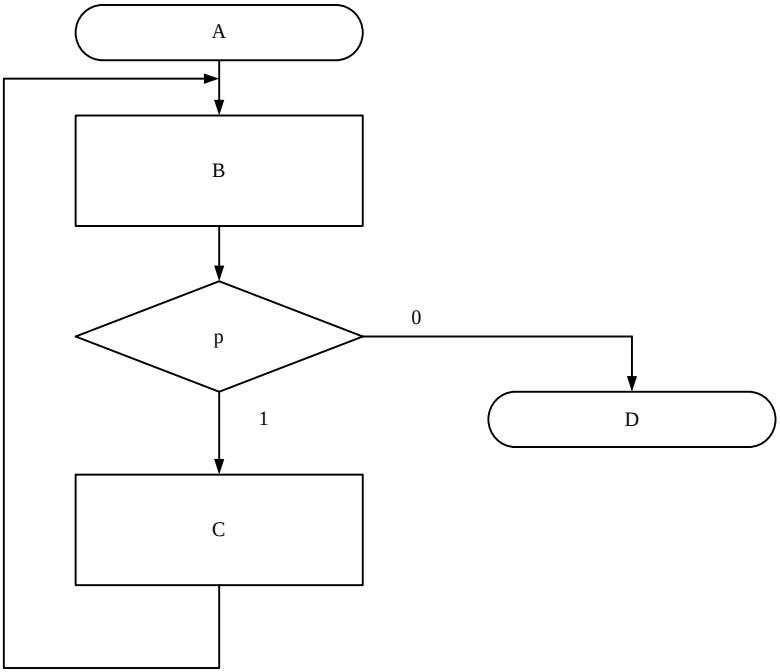



Рисунок 2.10 – Граф-схема алгоритму

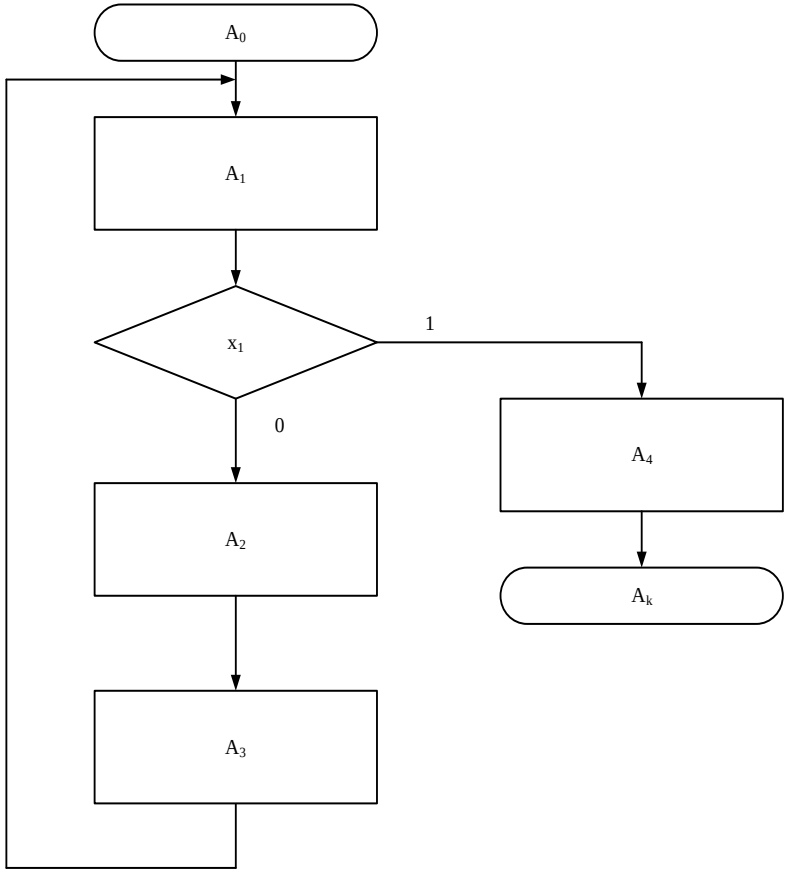



Рисунок 2.11 – Граф-схема алгоритму

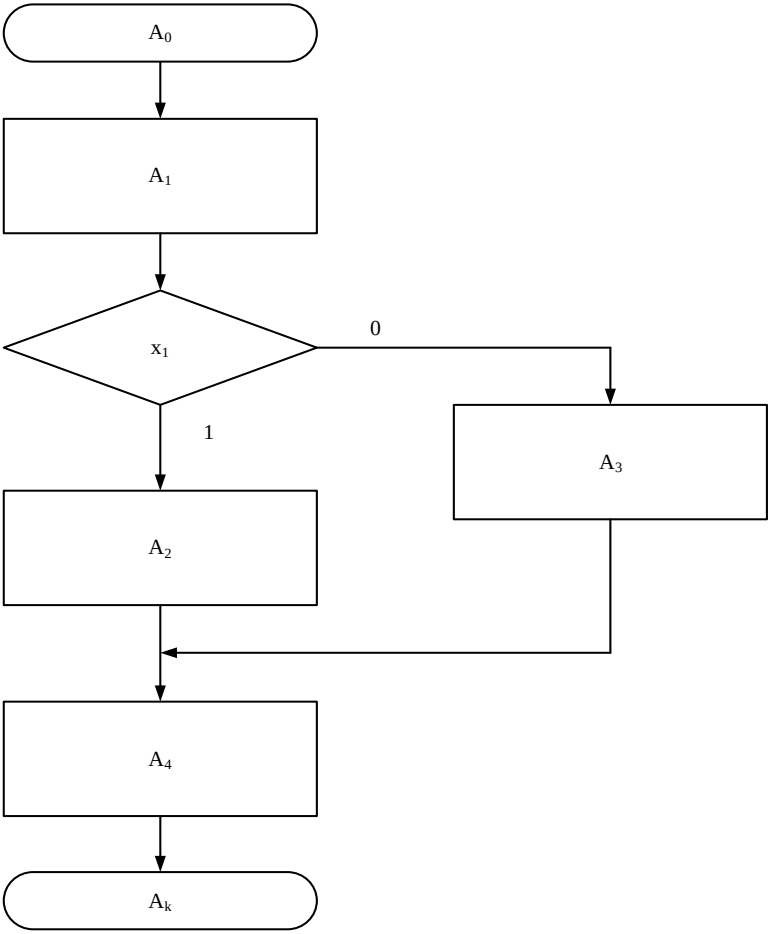



Рисунок 2.12 – Граф-схема алгоритму

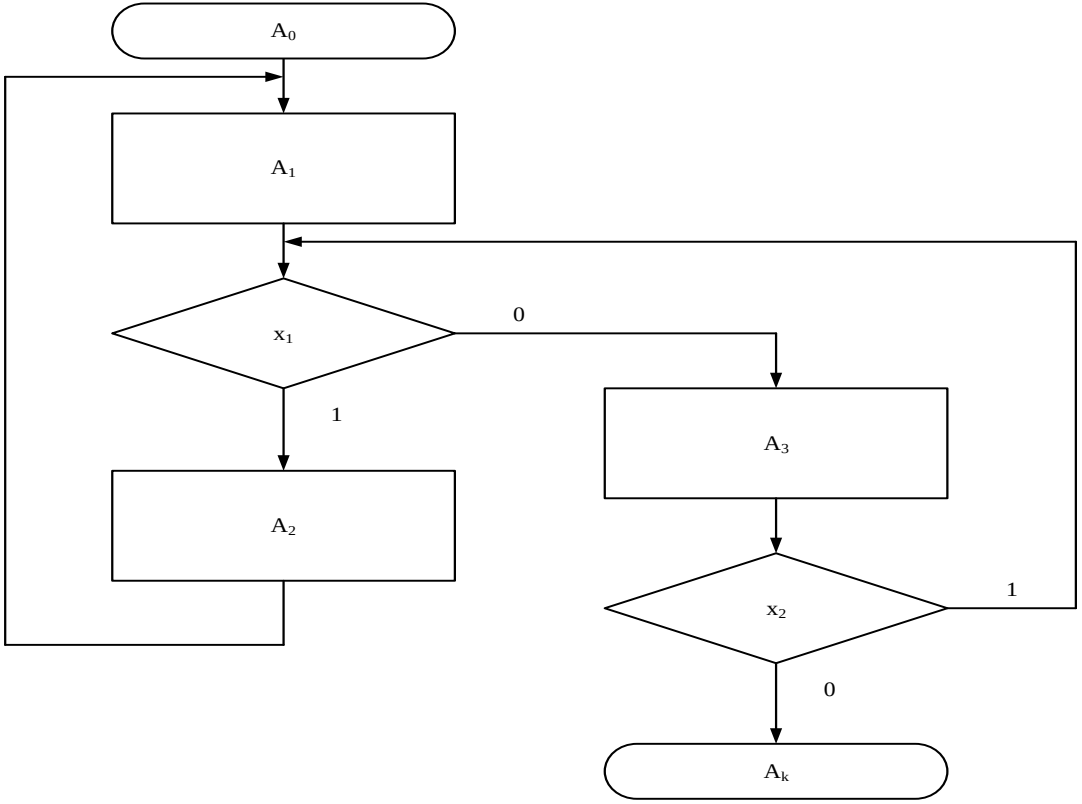



Рисунок 2.13 – Граф-схема алгоритму

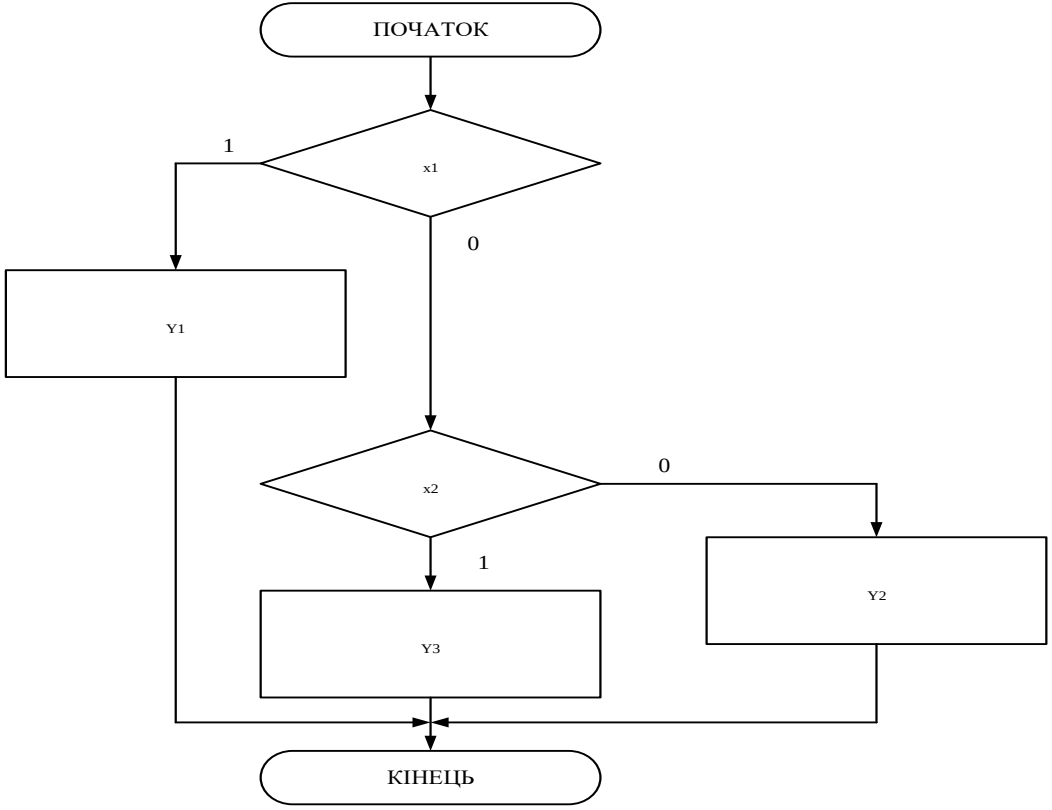



Рисунок 2.14 – Граф-схема алгоритму

## *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, цьому повинні виконуватися такі умови:

• відповідна ГСА має одну початкову та одну кінцеву вершини, а число операторних вершин дорівнює числу операторів у ЛСА;

• із ЛСА визначають оператор «Початок» і викреслюють відповідну йому початкову вершину ГСА;

• аналізують наступний за початковим оператор у ЛСА. Це може бути як керуючий оператор $A_i$, так і логічна умова $x_i$;

• вихід початкової вершини ГСА з'єднаний дугою із входом вершини, що відповідає в ЛСА найближчому розташованому праворуч від $A_i$ оператору або логічній умові. Аналогічно в ГСА проводять дуги з інших операторних вершин;

• одиничний вихід умовної вершини $x_i$ ГСА з'єднаний із входом вершини, розміщеної 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'єднаний із входом вершини (оператора або логічної умови), перед якою стоїть кінець стрілки з аналогічним номером;

• аналогічно будують наступні оператори ЛСА, з'єднують дужками, поки не будуть подані в ГСА всі оператори ЛСА;

• щоб переконатися в 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 прикладі $x_1 = 1$ передає керування оператору $A_2$ і в прикладі $x_1 = 0$ – оператору $A_3$. Будуємо вершини операторів $x_1$, $A_2$, $A_3$ на ГСА та 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 логічною є умова $x_2$, будуємо відповідний ромб, який повинен мати два виходи. У прикладі $x_2 = 1$ умовний оператор $x_2$ передає керування записаному після нього справа в ЛСА керуючому оператору $A_4$, відповідно креслимо дугу, що 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 випливає з того, що початок стрілки з номером «два» пов'язаний із 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.

У результаті вищенаведених операцій одержимо ГСА, зображену на рисунку 2.15.

***Завдання із зірочкою)))***

За поданими ЛСА побудувати ГСА:

1 $A_0 A_1 x_1 \uparrow^1 A_2 \downarrow^1 A_3 x_2 \uparrow^2 \downarrow^6 \downarrow^7 A_4 x_3 \uparrow^3 \omega_1 \uparrow^4 \downarrow^3 x_2 \uparrow^5 \omega_2 \uparrow^6 \downarrow^4 \bar{x}_4 \uparrow^7 A_5 A_6 \downarrow^2 \downarrow^5 A_7 A_k$

2 $A_0 \downarrow^2 A_1 x_1 \uparrow^1 x_2 \uparrow^2 \downarrow^1 A_2 x_3 \uparrow^3 \omega_1 \uparrow^4 \downarrow^3 x_1 \uparrow^5 x_2 \uparrow^6 \omega_2 \uparrow^7 \downarrow^4 A_3 \downarrow^5 x_4 \uparrow^8 \downarrow^6 A_4 \downarrow^7 \downarrow^8 A_5 x_1 \uparrow^9 A_6 \downarrow^9 A_7 A_k$

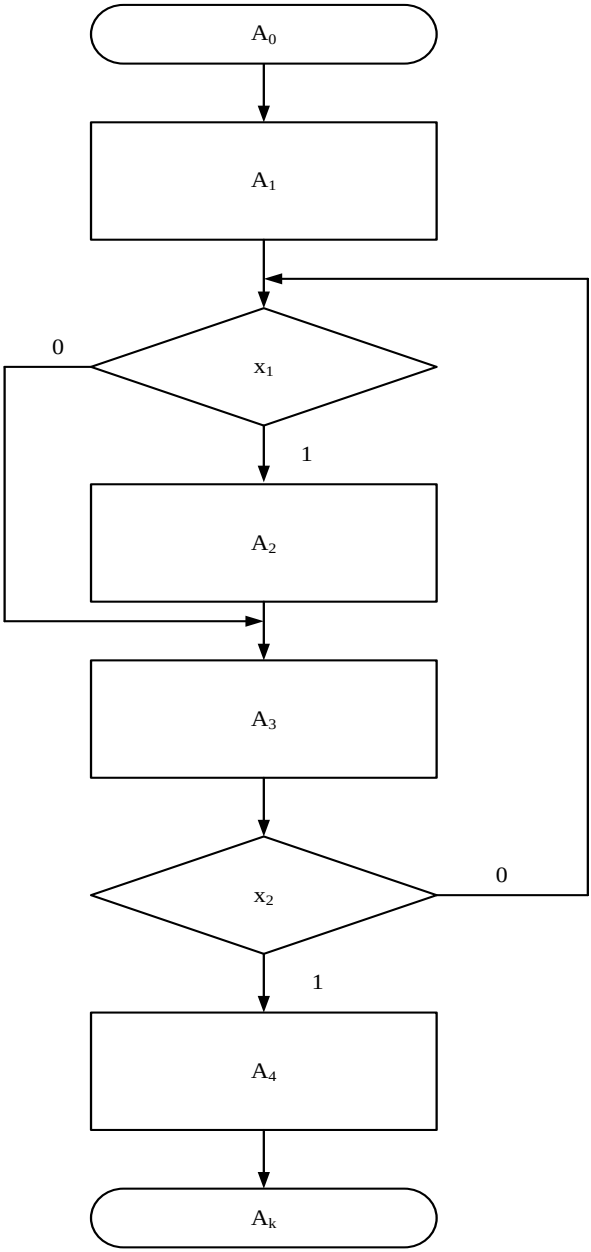



Рисунок 2.15 – Граф-схема алгоритму, побудована за заданою ЛСА

## 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* набуває значення *0,1,2, ..., к-1, j = 1,2, ..., к* є логічні функції переходу від операторів, що відповідають рядку до операторів, що відповідають стовбцям:

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

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

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

Якщо функція переходу $\alpha_{ij}$ , *i = 0,1,2, ..., к-1, j = 1,2, ..., к* 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 у матрицю можна не заносити.

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

Якщо який-небудь керуючий оператор $A_j$ виконується після керуючого оператора $A_i$ за певних значень логічних умов, то у відповідній клітині МСА (на перетині рядка з оператором $A_i$ та стовпця з оператором $A_j$) проставляють кон'юнкцію логічних умов за певних значень яких здійснюється перехід. Водночас символ логічної умови записують у 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 за умови декількох можливих комбінацій значень логічних умов, то відповідна функція переходу є диз'юнкцією кон'юнкцій логічних умов і їх заперечень, кожна з яких відповідає одній із 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 в 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.

Отже, 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 *к* – операторів в рядку повинна 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 лише 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 виконуватися хоча б один з операторів $A_j$, $j = 1,2, ..., к$.

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

$$x_1 \vee \overline{x_1}\,\overline{x_2} \vee \overline{x_1}\,x_2 =$$

$$= x_1 \vee \overline{x_1}\,(\overline{x_2} \vee x_2) = x_1 \vee \overline{x_1}(1) = x_1 \vee \overline{x_1} = 1$$

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

Якщо побудована матриця задовольняє обидві умови, значить матрична схема алгоритму побудована правильно.

***Означення.*** Логічну суму функцій переходів одного рядка називають *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: $Y_0 \to x_1 \vee \overline{x_1}\,\overline{x_2} \vee \overline{x_1}\,x_2$

$$Y_1 \to 1$$
$$Y_2 \to 1$$
$$Y_3 \to 1.$$

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

На практиці МСА широко використовують для зменшення кількості операторів у ГСА методом їх оптимізації. Тому спочатку здійснюють перехід від ГСА до МСА, потім за МСА проводять мінімізацію кількості операторів у ГСА без зміни функцій поданого в ГСА алгоритму й тоді здійснюють перехід до нової ГСА з меншою кількістю операторів.

### *1. Побудова ГСА за МСА*

1. Зображують початкову вершину ГСА, у якій проставляють оператор початку $A_0, i = 0$.

2. Проводять аналіз функцій переходів $\alpha_{ij} = \alpha_{0j}$, записаних у відповідному операторі $A_0$ рядка матриці. Якщо одна із 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 та 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. Будують вершину для оператора $A_j$, у яку проставляють її символьне позначення, і проводять з'єднання її входу із виходом оператора $A_0$.

4. У разі, якщо в одній із клітинок рядка оператора $A_0$ функцією переходів є змінна $x_\gamma, \gamma = 1, 2, \ldots, p,$ то будують вершину умовного оператора, яка має один вхід. Його

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

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

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

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

## *2. Побудова МСА за ГСА*

1. У результаті аналізу ГСА складається квадратна матриця МСА. Для цього рядки матриці позначають символами операторних вершин за винятком оператора $A_k$ «кінець». Перший верхній рядок матриці позначають символом $A_0$ «початок». Стовпці матриці позначають символами операторних вершин, починаючи з оператора $A_1$. Крайній стовпець позначають 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, зокрема які проходять через логічні (умовні) вершини.

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

а) якщо вершина $A_i$ та 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 $A_i$ та деякою іншою $A_j$ проходить через 1(0) вихід в умовній вершині, то у функцію переходу входить символ умовної вершини в 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 $A_i$ та 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 упевнитися, що виконуються перша й друга умови МСА

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

де $j \neq l$ , і

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

## Лекція 11. Перетворення логічної схеми алгоритму на матричну схему алгоритму й навпаки

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

### *1. Побудова МСА за ЛСА*

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

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

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

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

*b*) 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 - $\alpha_{ij} = 0$;

*c*) якщо деякий оператор $A_j$ виконується після $A_i$ за певною комбінацією значень логічних умов, то відповідна функція переходів записують як кон'юнкція тих логічних умов, за певних значень яких здійснюється перехід до керуючого оператора, водночас символ логічної умови записують у кон'юнкції в прямому вигляді, якщо перехід здійснюють за одиничного значення логічної умови, і в інверсному, якщо перехід здійснюють за нульовим значенням логічної умови.

4. Функції переходів, одержані відповідно до пункту 3, записують у рядок матриці, відповідний оператору $A_i$.

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

6. Здійснюють перевірку побудованої МСА на відповідність її першій і другій логічним умовам для МСА:

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

де $j \neq l$, і

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

Для цього складають логічні функції рядків і для кожної з них здійснюють перевірку на відповідність логічним умовам.

## *2. Побудова ЛСА за МСА*

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

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

2. Із системи формул перетворення виділяють формулу, відповідну оператору $A_i$, починаючи з $A_0$

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

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

*а*) якщо одна з функцій переходів $\alpha_{ij} = 1$, а інші дорівнюють нулю, то в цьому разі 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 подані логічними добутками, то для подальшого запису ЛСА аналізують перший зліва член (кон'юнкцію) формули перетворення, у якому знаходиться керуючий оператор $A_i$, що не траплявся раніше в ЛСА, що будується, і записують усі логічні змінні цього члена із запереченням або без, кожна з яких праворуч забезпечується початком пронумерованої стрілки, а потім записують оператор $A_i$, що входить до цього члена формули.

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

4. Зазначену процедуру виконують доти, доки всі оператори й логічні умови не увійдуть в ЛСА, що формується.

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

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

***Завдання із зірочкою)))***

1. За поданою МСА записати ЛСА та ГСА

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_к$ |
|---|---|---|---|---|---|---|
| $A_0$ | 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 проводять перетворення цих моделей на МСА, і потім виконують їх мінімізацію за правилами, наведеними нижче.

1. Будують МСА та перевіряють на несуперечність.
2. За МСА складають систему формул перетворень.
3. У кожній формулі перетворення знаходять змінні, що входять до всіх членів формул.
4. Шукають змінні, які є спільними для більшості формул перетворення.
5. Ці змінні виносять за дужки так, щоб одержати вираз типу $(x_i A_k \vee \bar{x}_i A_l)$. Такі вирази називають *елементарними дужками*. Оскільки в МСА може бути декілька варіантів винесення за дужки, то одержують систему дужкових виразів.
6. Аналізують одержану систему дужкових виразів. Якщо є однакові вирази в елементарних дужках, то залишають лише один, той що зустрівся першим. Інші такі самі вирази замінюють тотожними логічними умовами $\omega\uparrow^i$, кінці стрілок від яких займають місце перед виразом, що залишився в дужках, у одній із формул перетворення.
7. За одержаними формулами перетворення будують еквівалентну мінімальну 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.

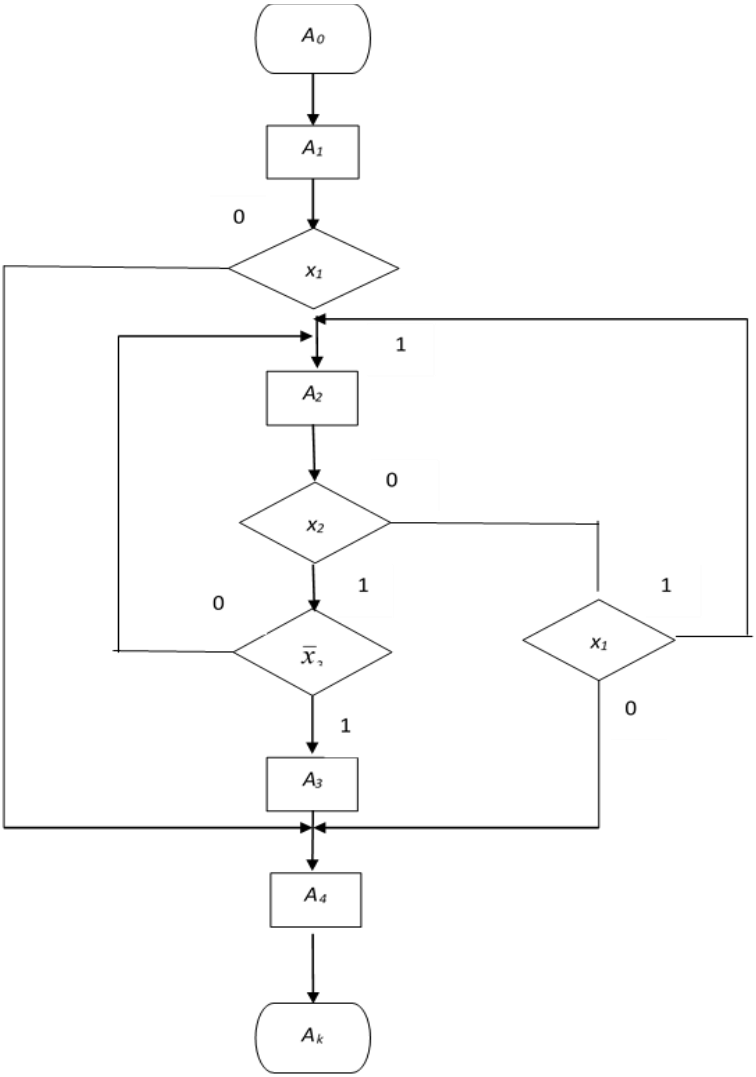



Рисунок 2.16 – Початкова ГСА

За ГСА побудуємо матричну схему та перевіримо її на несуперечність

| | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_к$ |
|---|---|---|---|---|---|
| $A_0$ | 1 | | | | |
| $A_1$ | | $x_1$ | | $\bar{x}_1$ | |
| $A_2$ | | $x_2x_3 \vee \bar{x}_2x_1$ | $x_2\bar{x}_3$ | $\bar{x}_2\bar{x}_1$ | |
| $A_3$ | | | | 1 | |
| $A_4$ | | | | | 1 |

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

$$A_0 \to A_1;$$
$$A_1 \to x_1A_2 \vee \bar{x}_1A_4 = \downarrow x_1A_2 \vee \bar{x}_1A_4$$
$$A_2 \to x_2x_3A_2 \vee \bar{x}_2x_1A_2 \vee x_2\bar{x}_3A_3 \vee \bar{x}_2\bar{x}_1A_4 =$$
$$= x_2(x_3A_2 \vee \bar{x}_3A_3) \vee \bar{x}_2(x_1A_2 \vee \bar{x}_1A_4) =$$
$$= x_2(x_3A_2 \vee \bar{x}_3A_3) \vee \bar{x}_2\omega\uparrow;$$
$$A_3 \to A_4;$$
$$A_4 \to A_к.$$

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

ЛСА, що відповідає спрощеній ГСА має такий вигляд:

$$A_0A_1 \downarrow^2 x_1 \uparrow^1 \downarrow^3 A_2x_2 \uparrow^2 \bar{x}_3 \uparrow^3 A_3 \downarrow^1 A_4A_k.$$

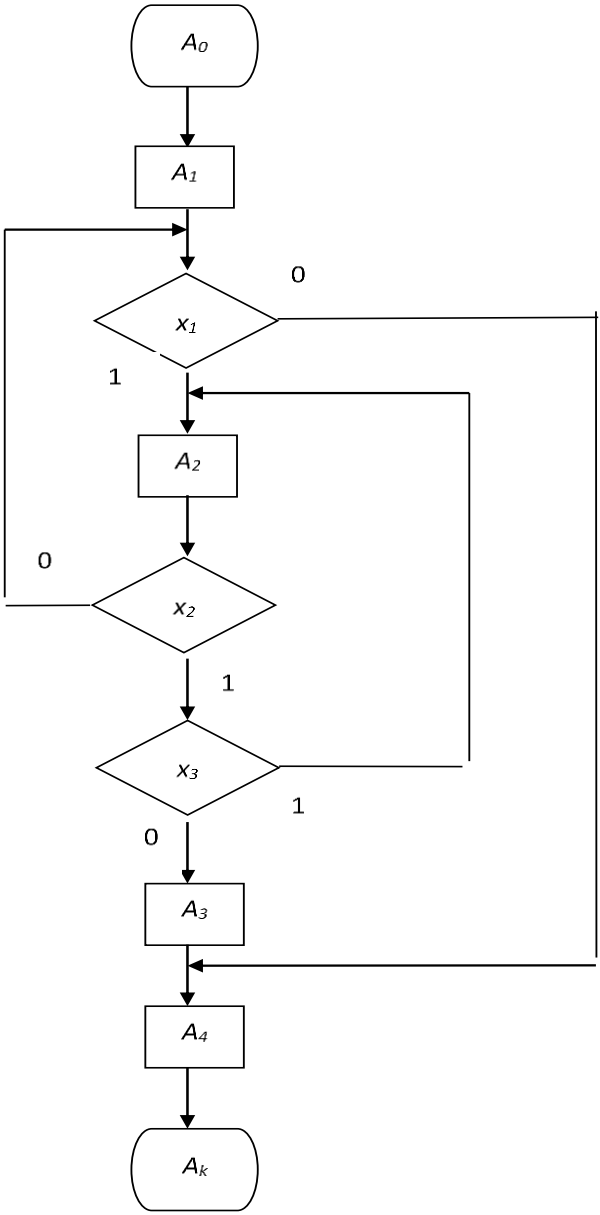



Рисунок 2.17 – Нова ГСА, побудована з урахуванням мінімізації алгоритму за кількістю логічних умов

***Завдання із зірочкою)))***

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

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

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

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

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

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

## Розділ ІІІ. Керуючі автомати зі схемною логікою

### Лекція 13. Керуючі автомати на комбінаційних схемах

На практиці широко використовують схемні рішення для побудови простих пристроїв керування. Водночас для характеристики станів керованого об'єкта, використовують набори двійкових змінних, одержаних від датчиків, перемикачів на пульті керування, кінцевих перемикачів та ін. Під час занесення в пристрій керування значень цих змінних їх об'єднують у набори, які відповідають тим чи іншим ситуаціям у керуючому об'єкті. Відповідно 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, у яких вводиться в дію та або інша логічна функція. Зручною формою їх задання є та, що ґрунтується 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 результаті 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 істотними. Замість прочерків може бути поставлено 1 або 0, але це ніяк не повинно впливати на дії в результаті.

Сукупність значень дій на всіх наборах змінних (умовах) має назву *вектору 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, у яке записують таблицю 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,* розміщених упоперек роликового транспортера. Пристрій керування має на основі сигналів датчиків $d_1$, $d_2$ та $d_3$, також розміщених упоперек транспортера, визначити розмір деталі та провести її сортування у відповідні нагромаджувачі деталей із заслінками $B_1$, $B_2$ та $B_3$, призначених для відбирання деталей кожного типу. Контроль розмірів деталей здійснюють датчиками $d_1$, $d_2$ та $d_3$.

Пристрій для сортування деталей за розміром наведено на рисунку 3.1.

***Розв'язування***. Можливі такі 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 $d_2$, $d_3$ й відповідно відкривається бункер $B_2$;

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

г) у разі, 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 символьна ГСА.

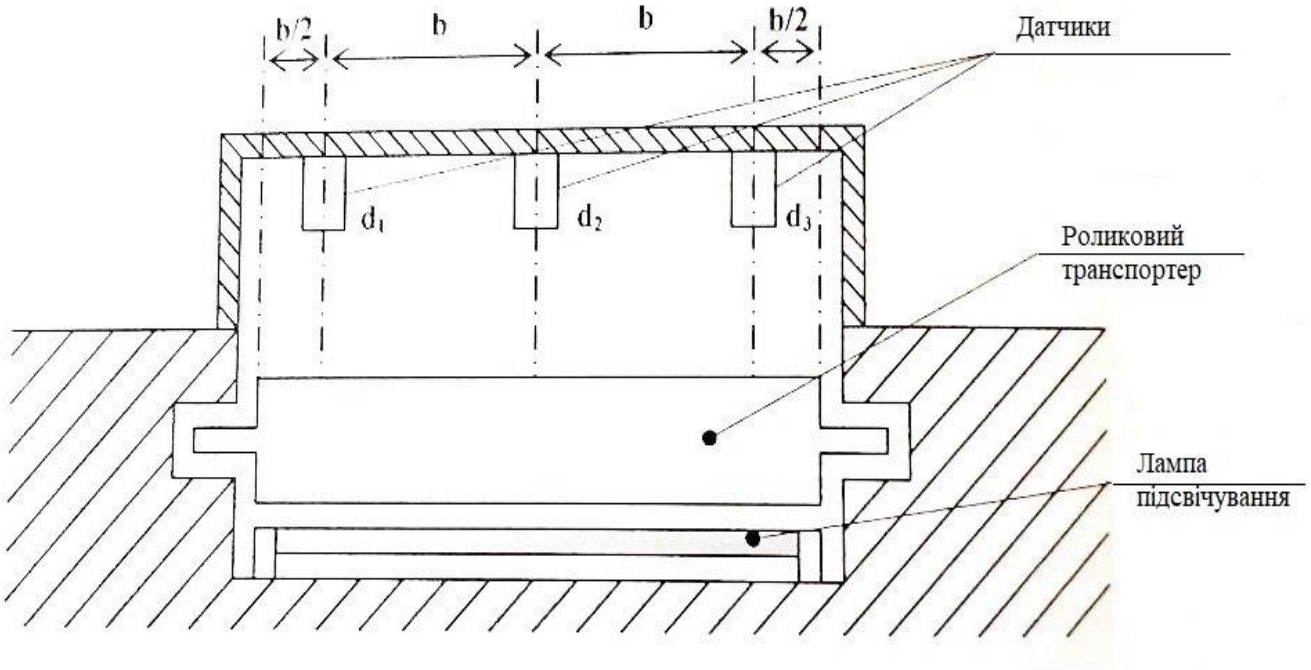



Рисунок 3.1 – Пристрій для сортування деталей за розміром

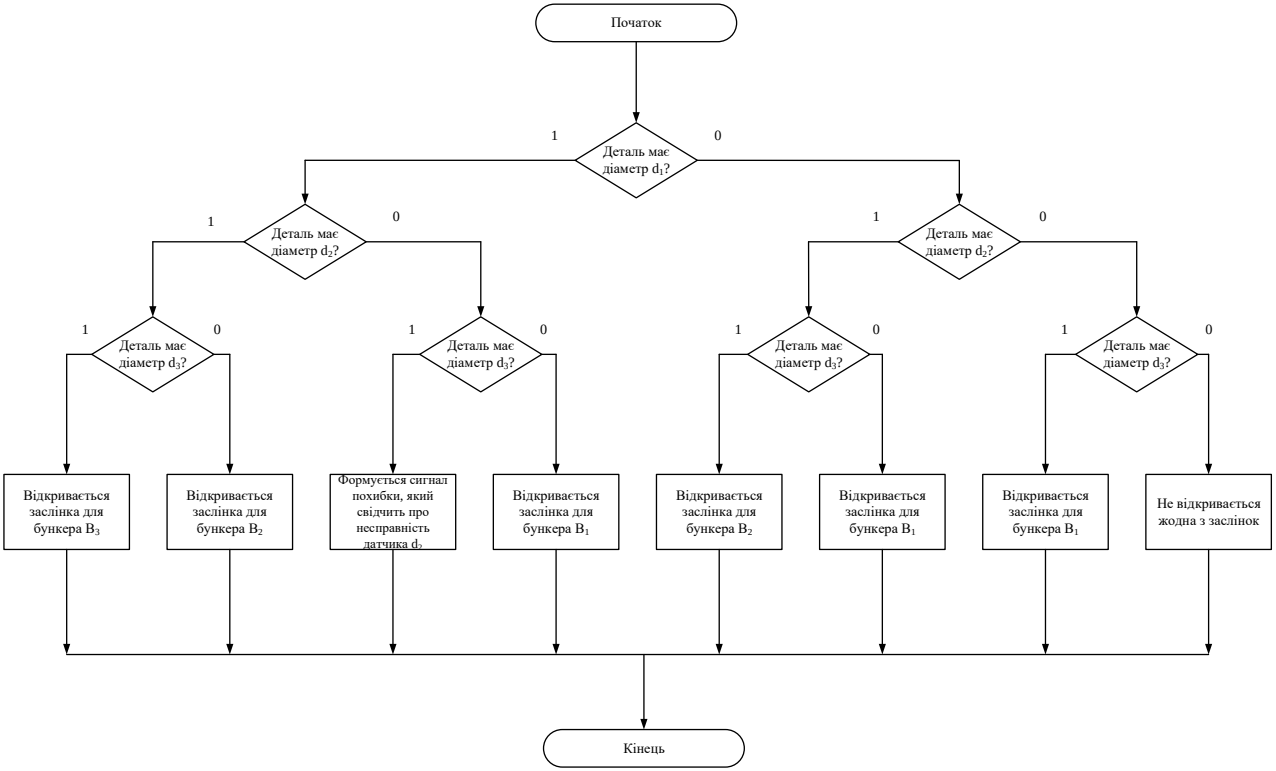



Рисунок 3.2 – Змістовна ГСА роботи керуючого автомата сортування деталей

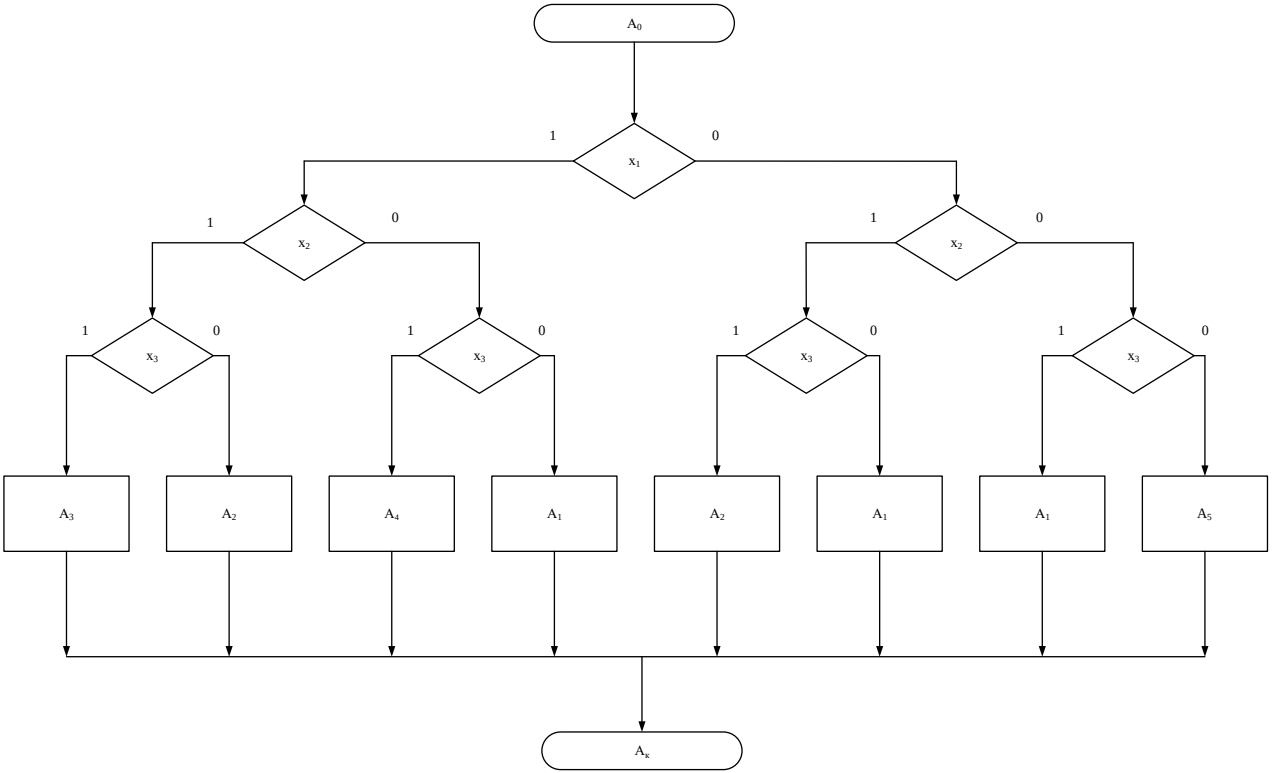



Рисунок 3.3 – Символьна ГСА керуючого автомата сортування деталей

За символьною 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 становлять 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.

Отже, 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, називають *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 проєктувальник EDSAC (Англ. Electronic delay storage automatic calculator – автоматичний обчислювач на електронних лініях затримки) – першого комп'ютера, що передбачає внутрішнє зберігання програм. Побудований у 1949, EDSAC використовував пам'ять на лініях затримки. Покоління ЕОМ 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, у якій уведено найважливіше поняття бібліотеки програм.

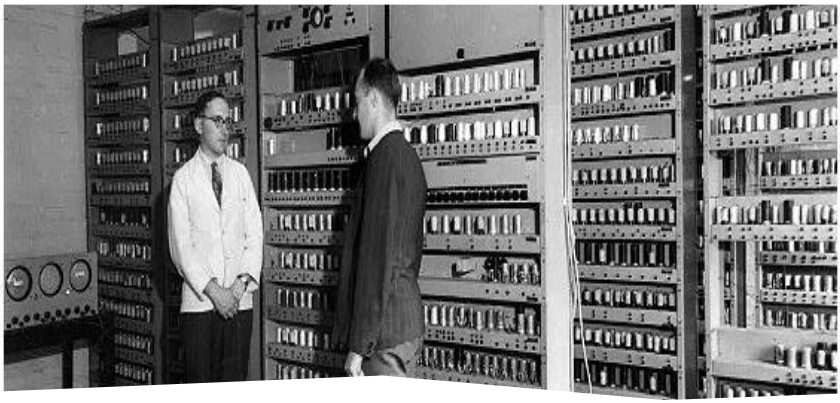


Рисунок 4.1 – Моріс Вілкс і Білл Ренвік зі своєю EDSAC

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

- Здобув освіту в коледжі короля Едуарда VI у Сторбриджі та в шкільні роки займався радіоаматорством, до якого його залучив учитель хімії.
- Навчався з 1931 по 1934 рік в Кембриджському університеті, де в 1936 році здобув 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 р. 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

центральний процесор у 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 символьних 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 називають багатокористувацькими операційними системами) та розподілених обчислень.
- У 1974 році Уїлкс зіткнувся зі швейцарською мережею передавання даних (у Hasler AG), яка використовувала кільцеву топологію для розподілу часу в мережі. Завдяки Уїлксу були сформовані комерційні товариства, а аналогічні технології стали широко доступними у Великобританії.

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

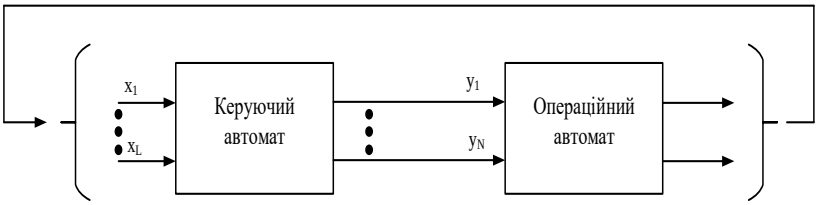



Рисунок 4.2 – Ілюстрація принципу «Розділяй і володарюй»

- Мікрооперації -МО – елементарні дії обробки інформації.
- Мікрокоманди -МК – набір МО, що виконують в одному такті.
- Мікропрограма – послідовність МК. Отже, 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 із жорсткою логікою для вирішення завдання керування широко застосовують мікропрограмні автомати різної міри універсальності та складності. Але загальним для них є наявність у їх структурі керуючої програми.

Найпростіші програмовані керуючі автомати, ґрунтуються на простому перебиранні керуючих команд. У цьому разі кожна подальша команда $A_{i+1}$ виконується після закінчення попередньої $A_i$, $i = 1, 2, \ldots, n$, 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, будучи послідовністю керуючих команд, і не має ні розгалужень, ні циклів (див. рис. 4.3).

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

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

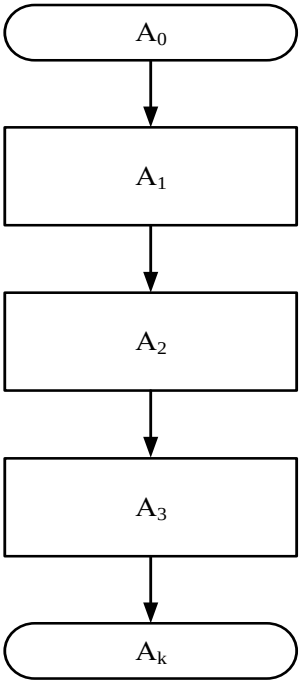



Рисунок 4.3 – Лінійна ГСА

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

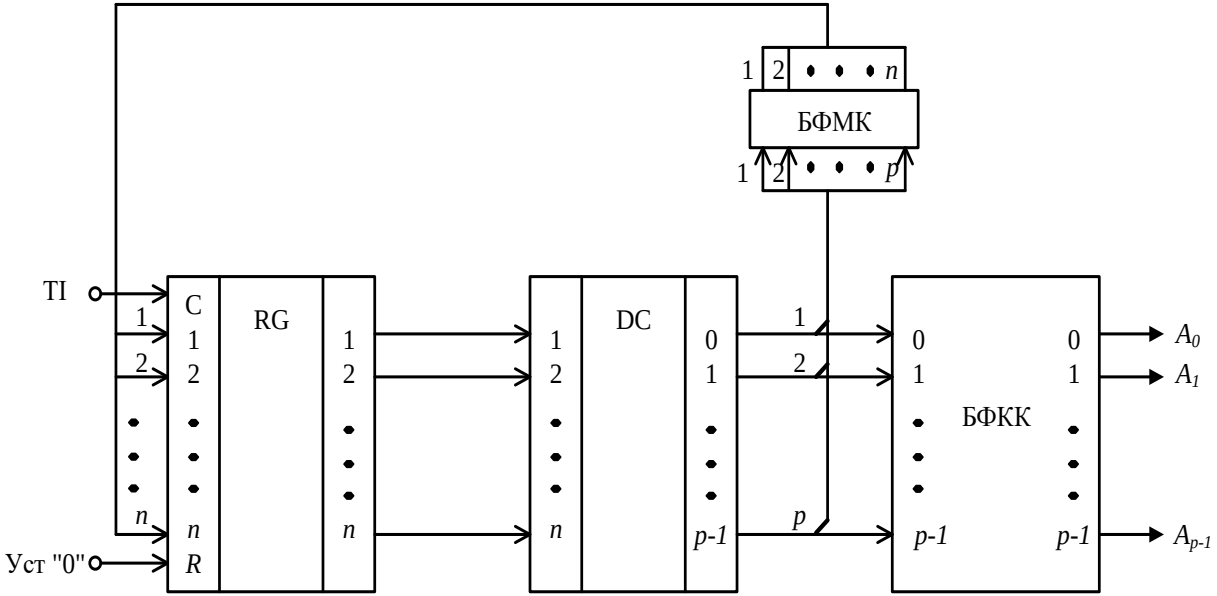



Рисунок 4.4 – Найпростіший керуючий МПА на регістрі

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

Блок формування команд керування вирішує завдання підсилення та 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 зі своїх $p$ входів в $n = ]log_2 p[$ виходів.

Працює МПА, шо розглядається, так: у початковому стані за відсутності тактового імпульсу ТІ регістр RG установлений сигналом Уст «0» у початковий нульовий стан. Відповідно на виході «0» дешифратора DC буде вироблятися сигнал, що надходить на відповідний йому вхід БФМК, де він перетворюється на двійкову кодову комбінацію мікрокоманди довжини $n$, яка потім подається на відповідні входи регістра RG. Одночасно із цим сигнал із дешифратора DC надходить також на відповідний йому вхід БФКК, де відбувається його формування 1 підсилення.

Під час надходження тактового імпульсу ТІ на синхровхід С регістра RG до нього заноситься нова мікрокоманда з БФМК, що повинна виконатися вслід за початковою мікрокомандою, а потім цикл роботи МПА повторюється відповідно до вищеописаного.

Розглянуту на рисунку 4.4 структуру МПА можна спростити, увівши замість регістра лічильник, наприклад, із 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 й БФКК.

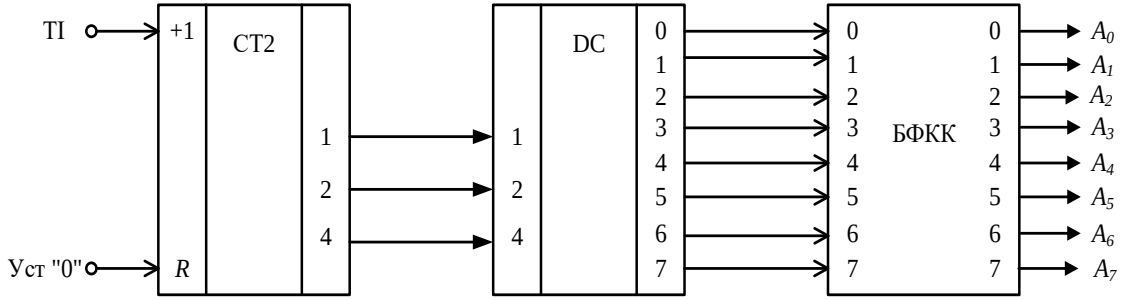



Рисунок 4.5 – Найпростіший керуючий МПА на лічильнику

У початковий нульовий стан керуючий автомат установлюється 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 DC і подаються на входи БФКК. У результаті на виходах БФКК послідовно з'являються керуючі команди $A_0, A_1, \dots, A_7$, що подаються на об'єкт керування до появи кінцевого сигналу $A_k = A_7$ про закінчення процесу керування.

МПА можна також реалізовувати за допомогою розподільників імпульсів, що набули поширення в різних керуючих пристроях унаслідок простоти своєї структури та доброї надійності, однак число тригерів, що міститься в них, дорівнює не $]log_2 p[$, як це було в МПА з регістрами чи лічильниками, а $p$, що за великої кількості команд керування призводить до додаткових апаратурних витрат.

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

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

Час перемикання МПА впливає на його швидкодію, а гонки – на його надійність. Тому час перемикання й гонок намагаються зменшувати, підвищуючи тим самим швидкодію та надійність МПА. Цього досягають застосуванням складних структур лічильників та розподільників імпульсів з протигоночними елементами та спеціальним кодуванням.

## Лекція 15. Універсальні мікропрограмні автомати Уїлкса

Розвиваючи структуру найпростішого керуючого мікропрограмного автомата, легко перейти до універсального мікропрограмного автомата, істотною відмінністю якого від найпростішого є те, що в ньому допускаються цикли й розгалуження в мікропрограмі, яка ним виконується.

**Означення.** Автомат, алгоритм якого заданий уведеною в нього мікропрограмою, що містить цикли й розгалуження, називають *універсальним мікропрограмним автоматом.*

Таким автоматом є автомат Уїлкса. Мікропрограмний автомат Уїлкса складається з регістра RG, дешифратора DC, блока перевірки логічних умов (БПЛУ), блока формування команд керування (БФКК) і блока формування мікрокоманд (БФМК), який є звичайним шифратором (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 в 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 І (див. рис. 4.7), на один із двох входів яких надходить сигнал $z(t)$ із дешифратора, що діє на момент часу $t$, а на інші входи цих схем – прямий та інверсний сигнали логічної умови $x(t)$ і $\overline{x}(t)$.

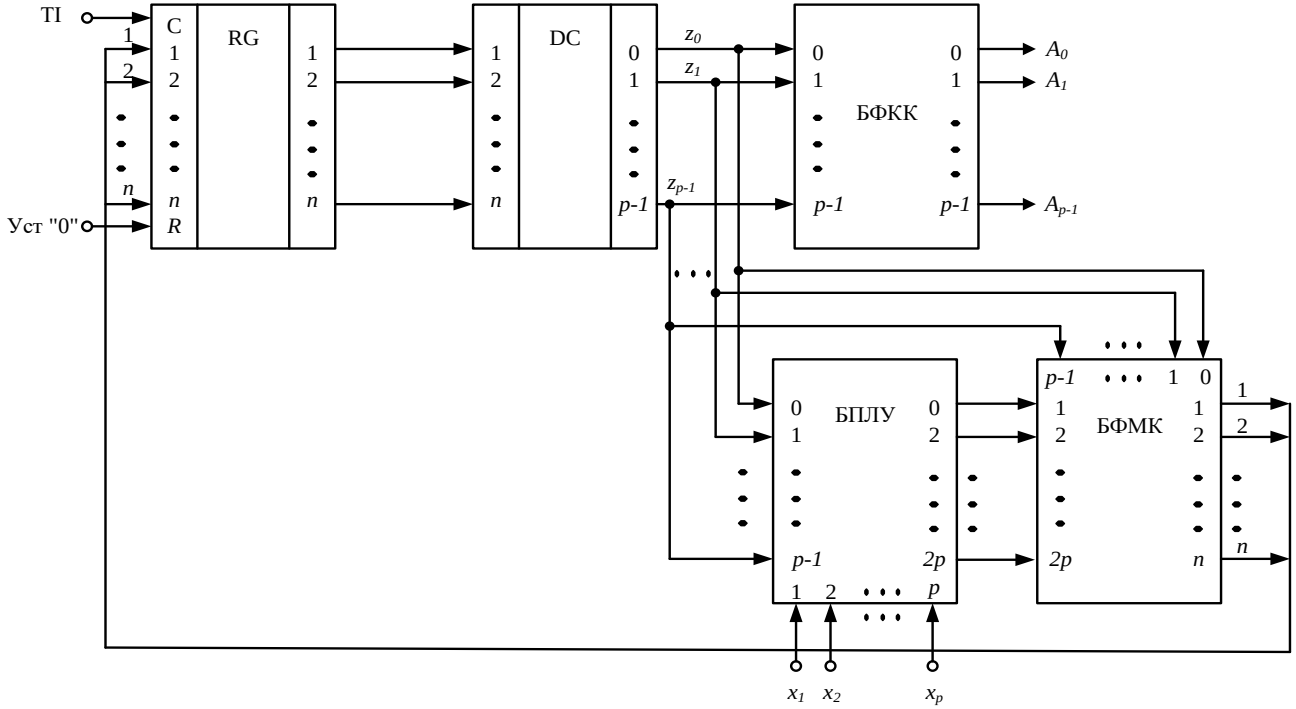



Рисунок 4.6 – Блок-схема мікропрограмного автомата Уїлкса

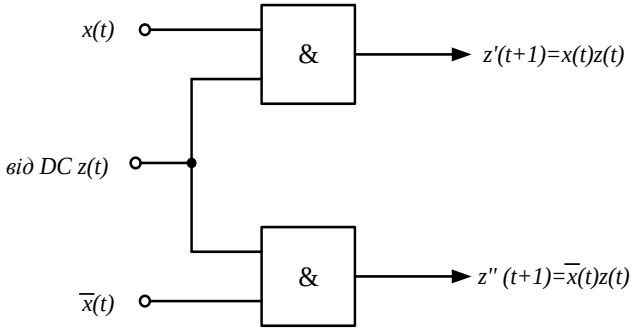



Рисунок 4.7 – Логічний вузол БПЛУ

На виходах схем І з'являється відповідно сигнал

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

або

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

який через БФМК задає мікрокоманду, яка буде виконана на 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 передбачене розгалуження в програмі, то відповідний логічний вузол буде відсутній і виконується наступна за порядком мікрокоманда, як це було показано в найпростішому мікропрограмному автоматі.

Якщо одна логічна умова пов’язана з другою, друга із третьою тощо, то $z'(t+1)$ і $z''(t+1)$ будуть визначатися добутком декількох логічних змінних. Можливо також, що в мікропрограмі здійснюється звернення до однієї й тієї самої мікрокоманди під час 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 задають 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, заданих 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 $z_0, z_1, \ldots, z_7$, кодування яких наведено в таблиці 4.1.

Ставлять задачу побудувати відповідний мікропрограмний автомат Уїлкса.

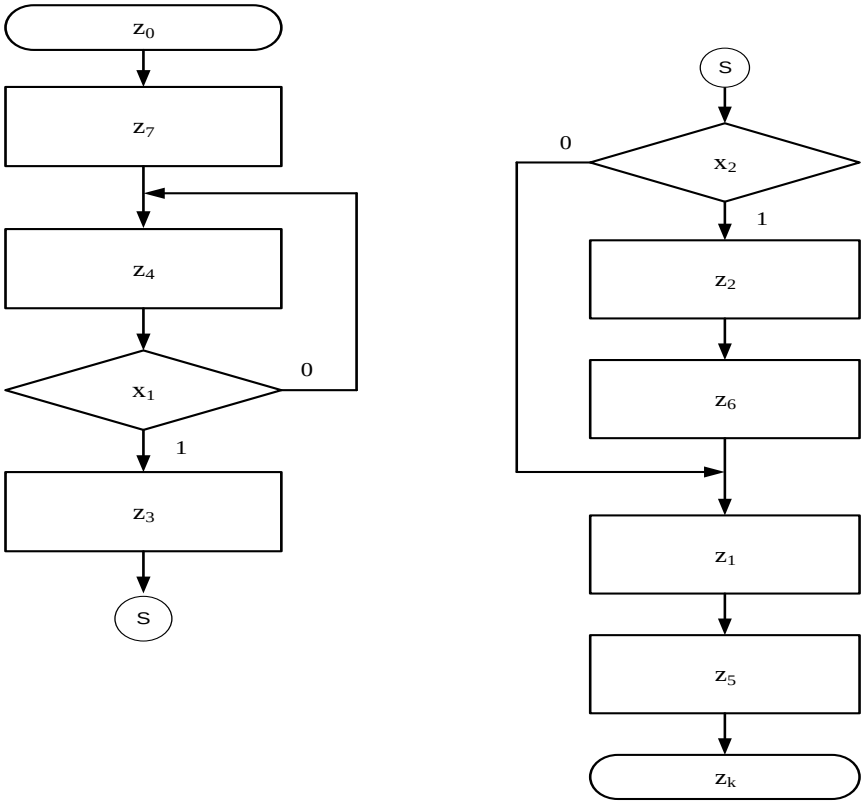



Рисунок 4.8 – Граф-схема алгоритму

Таблиця 4.1 – Кодування мікрокоманд

| № мікро-команди | $М_к$ | Код $М_к$ |
|---|---|---|
| 1 | $z_0$=$z_к$ | 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 з ГСА побудуємо кодовану мікропрограму, сформульовану так: визначаємо першу мікрокоманду на момент часу $t$ (табл. 4.2). Це буде мікрокоманда $z_0$ (початок). Із ГСА бачимо, що після неї на момент часу $t+1$ іде мікрокоманда $z_7$. Це означає, що з нульової шини дешифратора, яка відповідає мікрокоманді $z_0$, на всі схеми АБО шифратора (БФМК) повинні бути заведені зв'язки (рис. 4.8). У результаті на момент часу $t+1$ на виходах АБО шифратора з'являється двійкова комбінація 111, що надходить на входи $D1, D2, D4$ регістра RG і за ТІ, що надходить на вхід С на момент часу $t+1$, вносять у регістр. Після мікрокоманди $z_7$ аналогічно відпрацьовується мікрокоманда $z_4$. Після мікрокоманди $z_4$ іде логічна умова $x_1$. У разі, якщо вона дорівнює 1, то виконується команда $z_3$, у протилежному разі здійснюється повернення до команди $z_4$ і цикл замикається. В автоматі на рисунку 4.8 ця умова реалізується відповідним логічним вузлом, що складається з 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 $\overline{x_1}$. Як свідчить ГСА, вихід схеми, що відповідає добутку $z_4x_1$, необхідно з'єднати з першою та другою схемами АБО шифратора БФМК, що приводить під час виконання 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 І, на якому утворюється добуток $z_4\overline{x_1}$, повинен бути з'єднаний із входом третьої схеми АБО. У 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 в таблиці 4.2.

Таблиця 4.2 – Кодована мікропрограма

| № мікро-команди | *t* | *t+1* | Код $M_к$ |
|---|---|---|---|
| 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 | $z_5$ | $z_к = z_0$ | 0 0 0 |

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

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

Проведемо логічне об'єднання рівнянь, що мають у лівій частині однакові індекси. Це будуть за індексом чотири – друге й четверте та за індексом один – шосте й восьме рівняння:

1. $z_7^{t+1} = z_0^t$
2. $z_4^{t+1} = z_7^t \vee z_4^t \overline{x}_1^t$
3. $z_3^{t+1} = z_4^t x_1^t$
4. $z_2^{t+1} = z_3^t x_2^t$
5. $z_1^{t+1} = z_3^t \overline{x}_2^t \vee z_6^t$
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, що відповідає заданій ГСА (рис. 4.9).

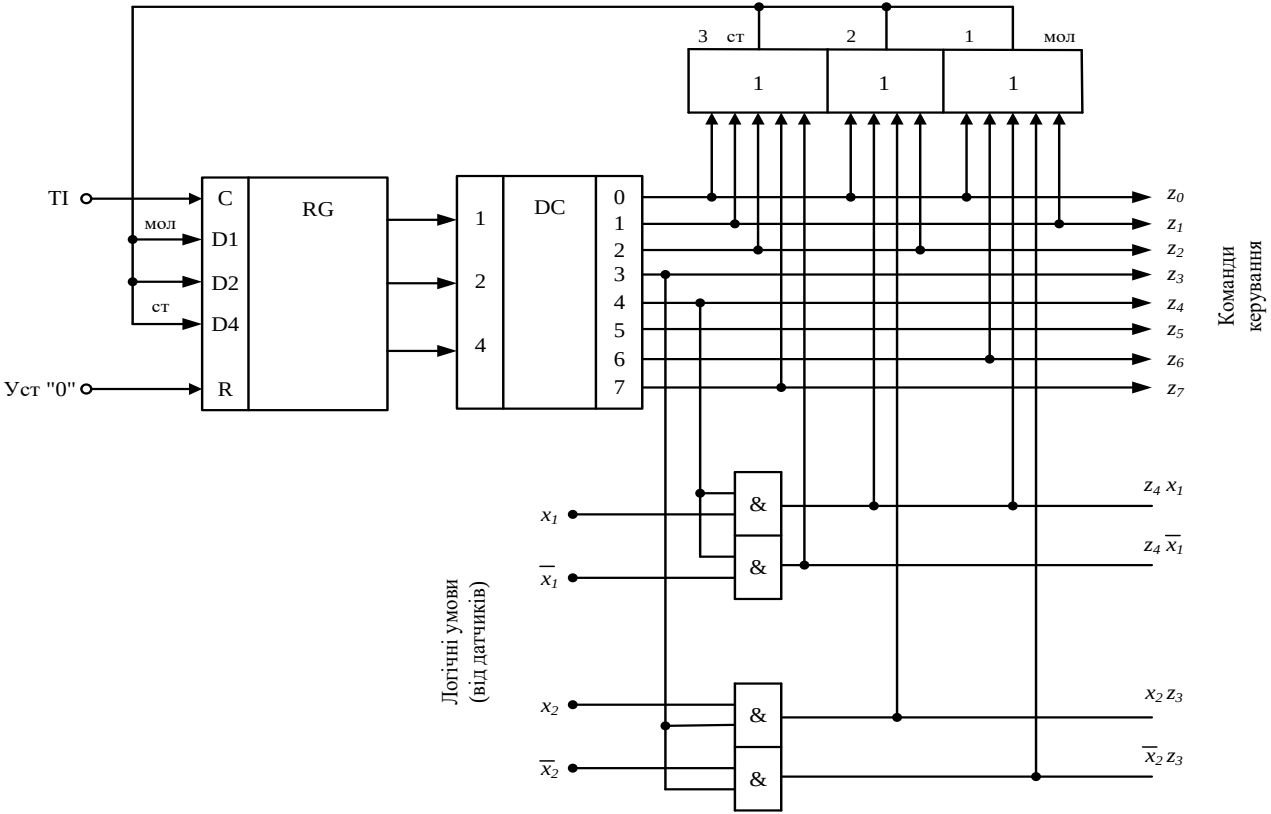



Рисунок 4.9 – Функціональна схема мікропрограмного автомата Уїлкса

# СПИСОК ЛІТЕРАТУРИ


1 Борисенко О. А. Керуючі системи : навчальний посібник. Київ : Центр навчальної літератури, 2004. 216 с.

2 Протасова Т. О., О. В. Д'яченко, О. А. Борисенко : методичні вказівки для лабораторних робіт із дисципліни «Керуючі системи». 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., Бабічева О. Ф. Аналіз, синтез і проектування цифрових систем керування : навч. посібник. Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. Харків : ХНУМГ ім. О. М. Бекетова, 2018. 150 с.

5 Білинський Й. Й., Книш Б. П. Цифрова схемотехніка. Електронно-обчислювальні пристрої : навчальний посібник. 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. Суми : Сумський державний університет, 2016. 200 с.

9 Борисенко О. А. Дискретна математика : підручник. 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»
всіх форм навчання

Відповідальний за випуск А. С. Опанасюк
Редакторка О. Ф. Дубровіна
Комп'ютерне верстання Т. О. Протасової

Формат 60x84/16. Ум. друк. арк. 5,95. Обл.-вид. арк. 5,18.

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