Тестування, верифікація та валідація ПЗ

Концидайло Андрій Михайлович

Quality Assurance Engineer
asp_kam1@student.ztu.edu.ua
@AndyFox96

Техніки тест дизайну

Тип тестування

Деякі тестування включають виконання компонента або системи, що тестується; таке тестування називається ***динамічним.*** *Інше тестування не передбачає виконання компонента або системи, що тестується; таке тестування називається* ***статичним тестуванням.***

1. Статичні техніки

(використовуються без виконання коду)

- **Рецензії (Reviews)** — перевірка документації, коду, тест-кейсів.
- **Інспекції (Inspections)** — формалізований аналіз артефактів із фіксацією дефектів.
- **Проведення walkthrough (ознайомчих переглядів)** — спільний розбір документа.
- **Аналіз (Static Analysis)** — автоматичне виявлення помилок у коді чи моделях.

Тестування чорної ящика

Тестування чорного ящика приймає масив вхідних даних і шукає генерацію визначених виходів. Ідея цієї назви полягає в тому, що вміст коду, що тестується, невідомий досліднику і, за визначенням, тестувальнику, який займається лише перевіркою функцій.

Тестування білого ящика

Тести білого ящика знаходяться на іншому кінці спектру. Вони засновані на точному знанні того, що відбувається з тестованим кодом, і тести виконуються насамперед для перевірки надійності коду, а не його абсолютної функціональності.

Тестування білого ящика виконується на початку процесу розробки за допомогою модульних тестів і на ранніх етапах фази інтеграції. Тестування чорного ящика є типовим для останніх етапів, де важлива реакція на конкретні сценарії роботи.

2. Динамічні техніки (на основі виконання коду)

2.1. Базовані на специфікації (Black-box)

(перевірка функціональності без знання внутрішньої логіки)

- **Еквівалентне розбиття (Equivalence Partitioning)**
- **Аналіз граничних значень (Boundary Value Analysis)**
- **Таблиця прийняття рішень (Decision Table Testing)**
- **Тестування переходів станів (State Transition Testing)**
- **Комбінаторне тестування (Pairwise, n-wise)**
- **Cause-Effect Graphing (Граф причинно-наслідкових зв’язків)**
- **Use Case Testing (Тестування на основі сценаріїв використання)**

2.2. Базовані на структурі (White-box)

(перевірка внутрішньої логіки, коду, структури)

- **Покриття операторів (Statement Coverage)**
- **Покриття гілок (Branch/Decision Coverage)**
- **Покриття умов (Condition Coverage)**
- **Покриття шляхів (Path Coverage)**
- **Testing Loops (перевірка циклів)**

2.3. Базовані на досвіді (Experience-based)

(використання знань, інтуїції та досвіду тестувальника)

- **Ad-hoc Testing (Імпровізоване тестування)**
- **Exploratory Testing (Дослідницьке тестування)**
- **Error Guessing (Прогнозування помилок)**
- **Checklist-based Testing (Тестування за чеклістами)**
- **Session-based Testing (Тестування в сесіях)**

3. Інші практичні техніки

(часто застосовуються в реальних проектах разом з основними)

- **Combinatorial Testing (Pairwise, All-pairs)**
- **Negative Testing (Негативне тестування)**
- **Regression Testing Techniques (Smoke, Sanity, Impact-based)**
- **Risk-based Testing (Ризик-орієнтоване тестування)**
- **A/B Testing (для web та mobile продуктів)**
- **Mutation Testing (для оцінки ефективності тестів)**

Техніки тест дизайну

Техніки чорного ящика (black box, behavior-based, specification-based techniques)

- Тестові умови, тестові сценарії та тестові дані виходять з базису тестування, який може включати вимоги, специфікації, сценарії використання та user stories
- Тестові сценарії можуть використовуватися для визначення невідповідностей та відхилень між вимогами та реалізацією

- Еквівалентне розділення
- Граничні значення
- Діаграма переходу станів
- Таблиця прийняття рішень
- Діаграма сценаріїв використання
- Попарне тестування

Техніки білого ящика (white box techniques, glass box, structure-based)

- Тестові умови, тестові сценарії та тестові дані виходять з базису тестування, який може включати код, архітектуру, детальну архітектуру або будь-яке інше джерело інформації про структуру програмного забезпечення
- Вимірювання покриття засноване на елементах структури (код, інтерфейси і т. д.)
- Специфікації використовуються як джерело додаткової інформації для визначення очікуваних результатів тестових сценаріїв

- Тестування та покриття операторів
- Тестування та покриття умов

Техніки, що ґрунтуються на досвіді

- Тестові умови, тестові сценарії та тестові дані виходять з базису тестування, який може включати знання та досвід тестувальників, розробників, користувачів та інших зацікавлених осіб.

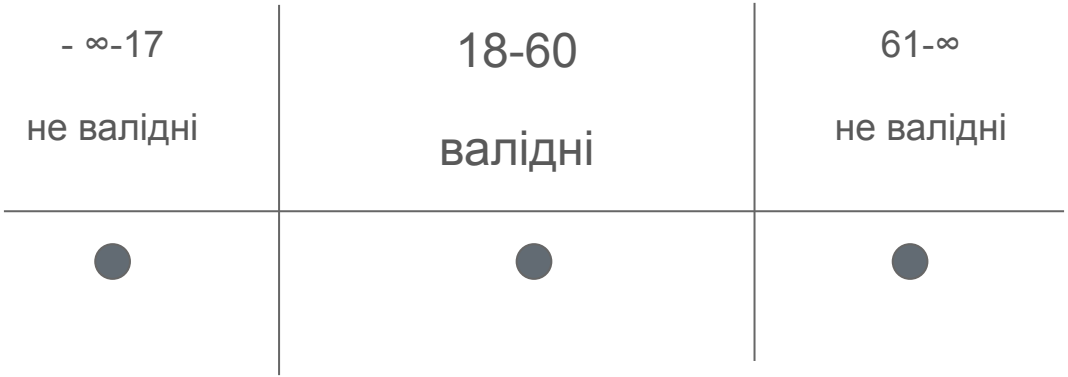
- Вгадування помилок
- Дослідницьке тестування
- Тестування на основі чек-лістів

Еквівалентне розділення

Еквівалентне розділення

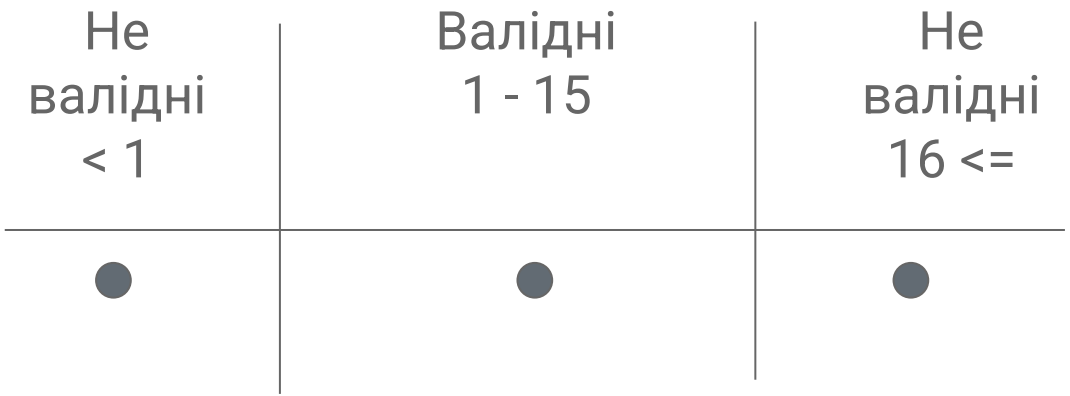
- Еквівалентне розбиття поділяє дані на групи (класи еквівалентності), які обробляються схожим способом
- Для досягнення 100% покриття за допомогою цього методу, тестові сценарії повинні покривати всі позитивні та негативні класи, перевіряючи хоча б одне значення кожного класу.
- Тобто в еквівалентному поділі з 1 діапазону створюється 1 тест кейс.

Пояснювальна бригада



Питання з ISTQB #1

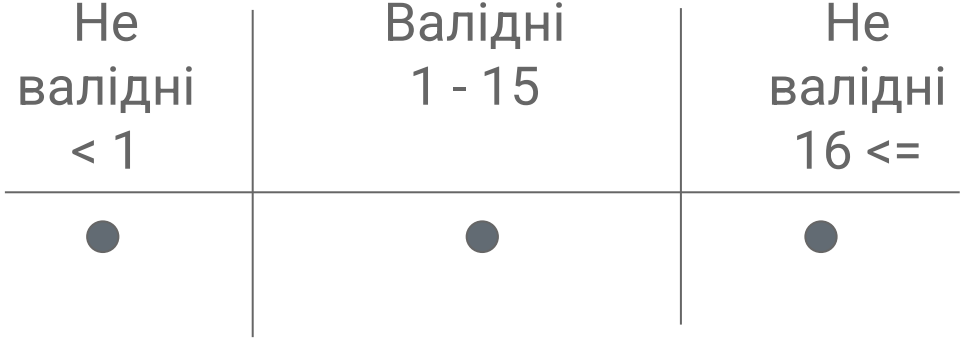
Числове поле має містити значення між 1 та 15. Використовуючи еквівалентний поділ, скільки мінімум тест кейсів потрібно створити для максимального покриття?



Питання з ISTQB #2

Числове поле має містити значення між 1 та 15. Використовуючи еквівалентний поділ, який із можливих варіантів є валідною колекцією еквівалентних класів у цьому сценарії?

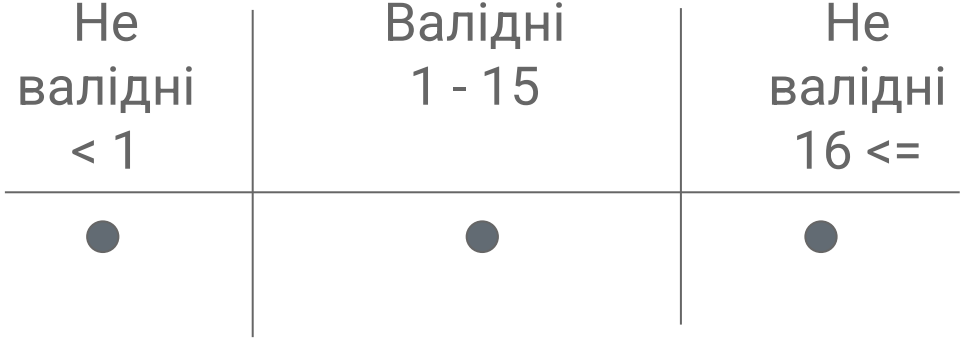
1. Менше 1, 1-15, більше 15
2. Негативні числа, 1-15, більше ніж 15
3. Менше 1, 1-14, більше 15
4. Менше 0, 1-14, 15 і більше



Питання з ISTQB #2

Числове поле має містити значення між 1 та 15. Використовуючи еквівалентний поділ, який із можливих варіантів є валідною колекцією еквівалентних класів у цьому сценарії?

1. **Менше 1, 1-15, більше 15**
2. Негативні числа, 1-15, більше ніж 15
3. Менше 1, 1-14, більше 15
4. Менше 0, 1-14, 15 і більше



Питання з ISTQB #3 (для фанатів пажощє)

У системі для розрахунку податку, який потрібно сплатити, у працівника 4000 зарплати не оподатковується. Наступні 1500 оподатковуються 10% податку. Наступні 28 тисяч оподатковуються 22% податку. Усі наступні суми оподатковуються 40%. Які з наступних груп зарплат потраплять до одного еквівалентного класу?

1. 4800, 14000, 28000
2. 5200, 5500, 28000
3. 28001, 32000, 35000
4. 5800, 28000, 32000

Питання з ISTQB #3 (для фанатів пажощє)

У системі для розрахунку податку, який потрібно сплатити, у працівника 4000 зарплати не оподатковується. Наступні 1500 оподатковуються 10% податку. Наступні 28 тисяч оподатковуються 22% податку. Усі наступні суми оподатковуються 40%. Які з наступних груп зарплат потраплять до одного еквівалентного класу?

1. 4800, 14000, 28000
2. 5200, 5500, 28000
3. 28001, 32000, 35000
4. **5800, 28000, 32000**

Тут можна не тільки числа!

Валідні	Не валідні
+38099 999 99 99	+38099 999 99 99 9
099 999 99 99	+38099 999 99 9k
abcdefuuuuu@gmail.com	abc@gmail.com
88005553535@gmail.com	12345@gmail.com
i_love_rocknroll_duzhe_sylno@gmail.com	iloverocknrollgmail.com
	iloverocknroll@gmailcom

Граничні значення

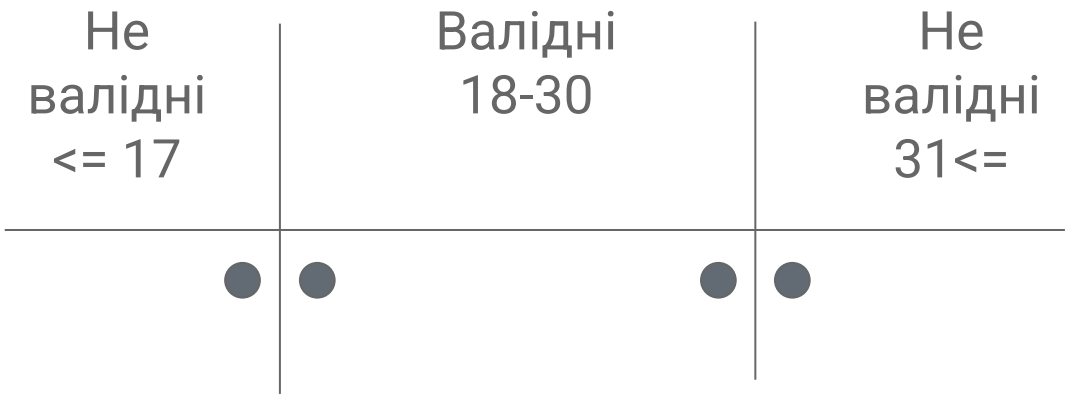
Еквівалентне розділення

- Аналіз граничних значень перевіряє межі числового діапазону
- Значення з валідного діапазону називаються валідні граничні значення, та якщо з невалідного діапазону - невалідні значення
- Граничні значення визначаються як ЛГ, ЛГ-1, ПГ, ПГ+1, де ЛГ та ПГ - це нижня та верхня межі у сценарії

Пояснювальна бригада

Питання з ISTQB #1

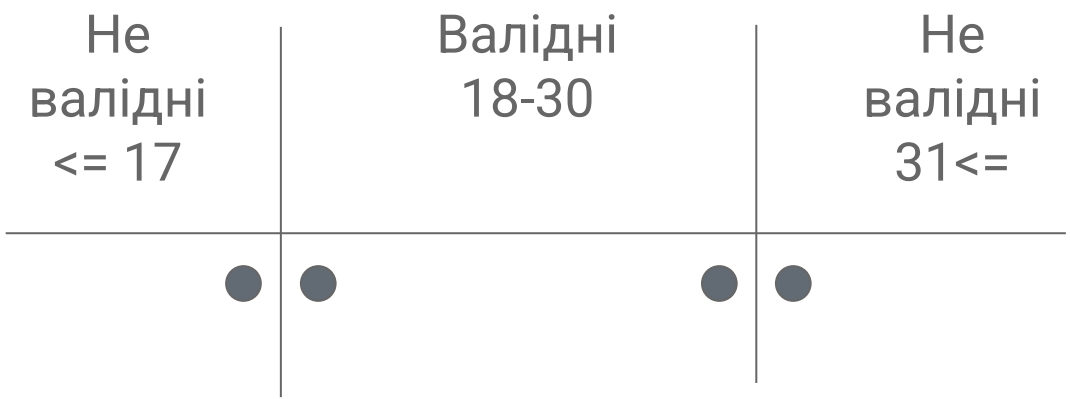
Текстове поле в програмі приймає введення віку користувача. Значення від 18 до 30, включаючи 18 і 30, будуть прийняті системою. Використовуючи техніку Граничних значень, яка мінімальна кількість тестів кейсів потрібна для максимального покриття?



Питання з ISTQB #2

Текстове поле в програмі приймає введення віку користувача. Значення від 18 до 30, включаючи 18 і 30, будуть прийняті системою. Використовуючи техніку межових значень, який варіант відповіді містить валідну колекцію граничних значень?

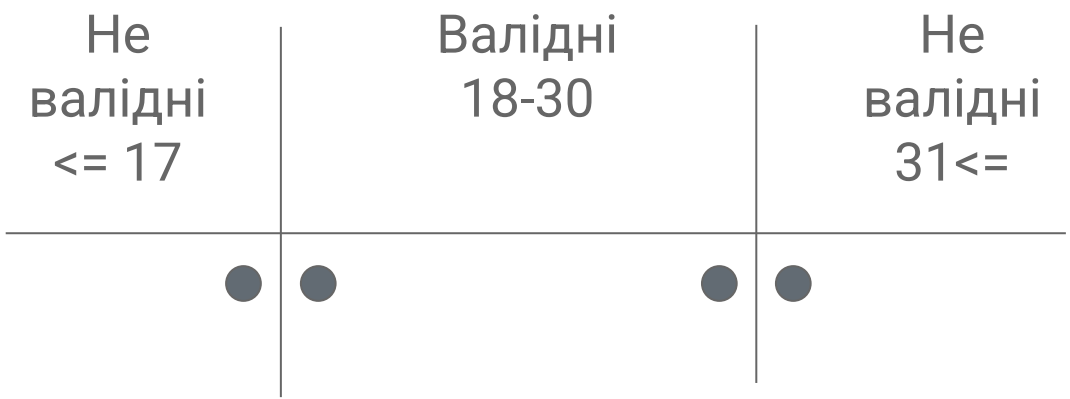
1. 16, 17, 19, 30
2. 17, 18, 19, 31
3. 17, 18, 30, 31
4. 18, 19, 20, 31



Питання з ISTQB #2

Текстове поле в програмі приймає введення віку користувача. Значення від 18 до 30, включаючи 18 і 30, будуть прийняті системою. Використовуючи техніку межових значень, який варіант відповіді містить валідну колекцію граничних значень?

1. 16, 17, 19, 30
2. 17, 18, 19, 31
3. **17, 18, 30, 31**
4. 18, 19, 20, 31



Питання з ISTQB #3

Текстове поле в програмі приймає введення віку користувача. Значення від 18 до 30, включаючи 18 і 30, будуть прийняті системою. Використовуючи техніку Граничних значень та Еквівалентного поділу, який варіант відповіді містить валідні граничні значення та валідне еквівалентне значення?

1. 17, 18, 20
2. 18, 30, 25
3. 18, 30, 31
4. 19, 20, 31

Питання з ISTQB #3

Текстове поле в програмі приймає введення віку користувача. Значення від 18 до 30, включаючи 18 і 30, будуть прийняті системою. Використовуючи техніку Граничних значень та Еквівалентного поділу, який варіант відповіді містить валідні граничні значення та валідне еквівалентне значення?

1. 17, 18, 20
2. **18, 30, 25**
3. 18, 30, 31
4. 19, 20, 31

Де можна використовувати цю техніку?

Всюди, де є числові обмеження:

- вік
- довжина паролю
- номер телефону

Навіщо потрібна ця техніка?

Для того, щоб люди не тестували 10 символів у паролі, які нікому не потрібні, тому що вони знаходяться на проміжку між 6 і 16. Для того, щоб не тестувати вік 20, 30, 40, тому що всі ці значення знаходяться між 18 і 60. **Тому що всі ці перевірки будуть безглузді, адже ми вже протестували граничні значення.**

Таблиця Прийняття рішень

Таблиця прийняття рішень

- Таблиця прийняття рішень покриває системні вимоги, у яких є логічні умови
- Специфікації аналізують та зображають Умови та Дії системи у формі таблиці
- Найчастіше Умови та Дії подаються у формі “True” та “False”.

Умови	**Тест кейс 1**	**Тест кейс 2**	**Тест кейс 3**	**Тест кейс 4**
Умова 1	True	True	False	False
Умова 2	True	False	True	False
Дії				
Дія	x	x	x	x

Який очікуваний результат для кожного з 2 тест кейсів?

	Правило 1	Правило 2	Правило 3	Правило 4
Умови				
Власник картки Сітібанк	Так	Так	Ні	Ні
Тип кімнати	Срібна	Платина	Срібна	Платина
Дії				
Покращити до Золотої	Так	Ні	Ні	Ні
Покращити до Срібної	N/A	Так	N/A	Ні

Тест кейси:

А. Власник картки, знімає Срібну

Б. Не має карти, знімає Платинову

Варіанти відповідей:

А - не покращувати, Б - не покращувати

А - не покращувати, Б - покращити до Золотої

А - покращити до Срібної, Б - покращити до Срібної

А - покращити до Золотої, Б - не покращувати

Дано наступну ТПР. Які з тест кейсів та очікуваних результатів ВАЛІДНІ?

	Правило 1	**Правило 2**	**Правило 3**	**Правило 4**
Умови				
Вік	< 21	21-29	30-50	> 50
Клас страхівки	A	A / B	B / C / D	C / D
Дії				
Преміум	100	90	70	70
Ультра	2500	2500	500	1000

1. 23 роки, клас A, Преміум- 90, Ультра - 2500
2. 51 рік, клас C, Преміум- 100, Ультра - 500
3. 31 рік, клас B, Преміум- 70, Ультра - 2500
4. 43 роки, клас C, Преміум- 100, Ультра - 1000

Створюємо свою Таблицю прийняття рішень

- Валідні варіанти ми позначатимемо словом true або буквою t, або цифрою 1.
- Невалідні ми позначатимемо або словом false або буквою f, або цифрою 0.

Чому цифри 1 та 0? Так пишуть програмісти. У бінарному коді: 0 – це брехня, 1 – це правда. Наприклад, якщо лампочка горить – це 1, а якщо вона вимкнена – 0.

Приклад №1

Потрібно завантажити аватарку, і є вимоги: вона має важити менше 1 Гб і бути у форматі фото (jpg, png або gif). У таблиці я роблю два рядки:

1. Чи це фото?
2. Чи підходить розмір?

$$2^2 = 4$$

	1	2	3	4
Формат jpg?	нет	нет	да	да
Розмір до 1 Гб?	нет	да	нет	да
Результат	Fail	Fail	Fail	Pass

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.																
ПІБ																
Нікнейм																
Пароль																

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ПІБ																
Нікнейм																
Пароль																

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ПІБ	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Нікнейм																
Пароль																

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ПІБ	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Нікнейм	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Пароль																

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ПІБ	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Нікнейм	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Пароль	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Приклад №2

На прикладі Instagram ми маємо заповнити чотири поля:

1. мобільний телефон чи електронну адресу;
2. ім'я та прізвище;
3. нікнейм користувача;
4. пароль

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Електронна адр.	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
ПІБ	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
Нікнейм	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Пароль	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Результат	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T

Приклад №3

В супермаркеті для залучення покупців власник вигадав систему знижок:

1. Якщо ти вперше в магазині, то маєш 10% знижки.
2. Якщо ти постійний клієнт, то у тебе є золота карта та 20% знижки.
3. Якщо у тебе сьогодні день народження, то тобі дають 50% знижки.
4. Якщо ти, скажімо, розбив там пляшку вина і втік, то тебе кидають у бан і нічого не будуть продавати

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Вперше	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Є знижкова карта	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
День народження	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Бан	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Результат	0	x	50	x	20	x	50	x	10	x	50	x	x	x	x	x

Попарне тестування

Попарне тестування

— це методика систематичної комбінаторики тестів, котра забезпечує ефективне зниження кількості тестів для перевірки реакції системи на можливі комбінації значень її вхідних параметрів.

Методика заснована на статистичному припущенні, що достатньо перевірити усі можливі значення пар вхідних параметрів для того щоб виявити більшість дефектів системи залежних від вхідних параметрів.

Головні цілі Pairwise Testing:

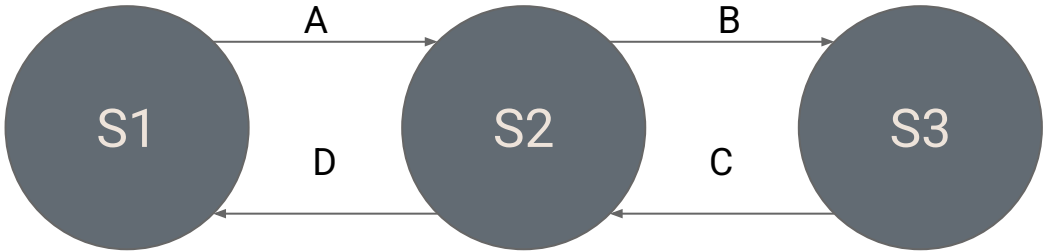
- прибрати надлишкові перевірки;
- забезпечити гарне тестове покриття;
- виявити найбільшу кількість багів на мінімальному наборі тестів.

http://pairwise.teremokgames.com/

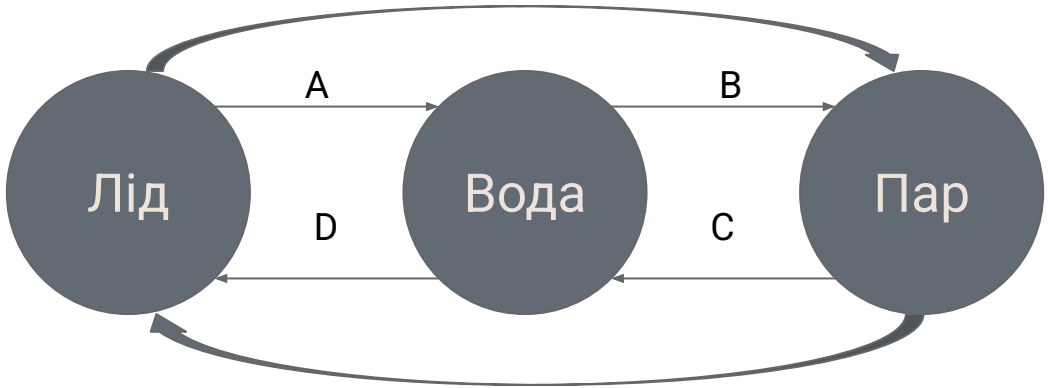
Діаграма переходу станів

Діаграма переходу станів

- Діаграма переходу станів показує початковий і кінцевий стан системи, а також описує переходи між станами.
- Діаграма переходу станів показує лише валідні переходи.
- Діаграма складається з пар переходів між двома станами.
- Якщо переходу між двома станами немає, то перехід вважається НЕвалідним.



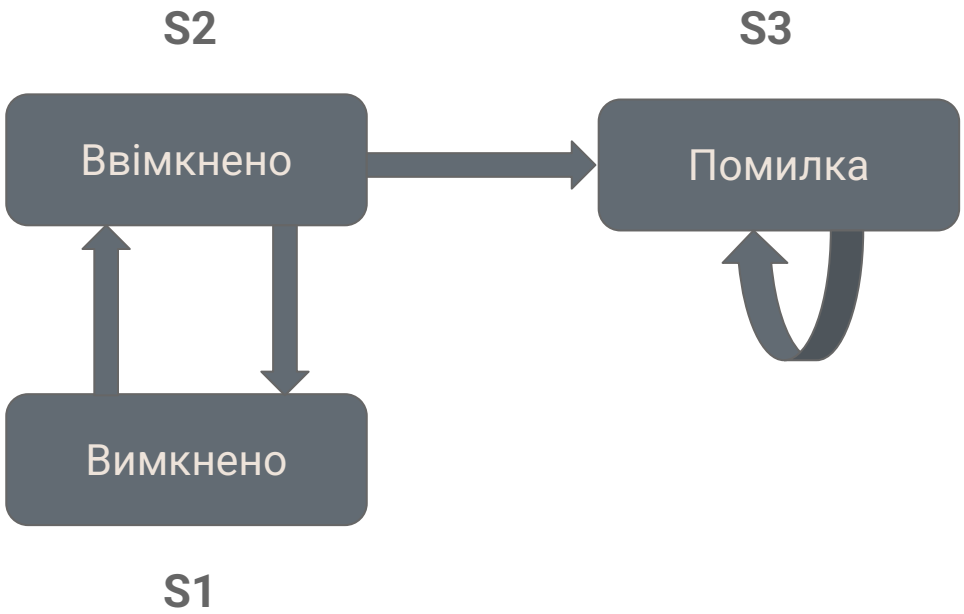
Діаграма переходу станів



Test Case ID	TC01	TC02	TC03	TC04
Початковий стан	Лід	Вода	Вода	Пар
Перехід	A	D	B	C
Фінальний стан	Вода	Лід	Пар	Вода

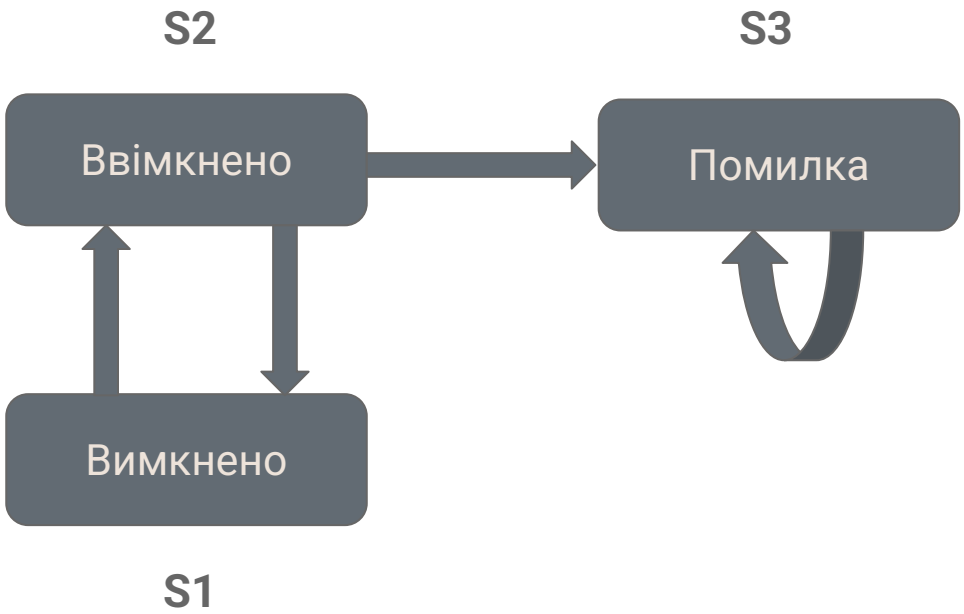
Грунтуючись на діаграмі переходу станів вмикача, який тест невалідний?

- Вимкнено-> ввімкнено
- Ввімкнено -> вимкнено
- Помилка -> ввімкнено
- Ввімкнено -> помилка



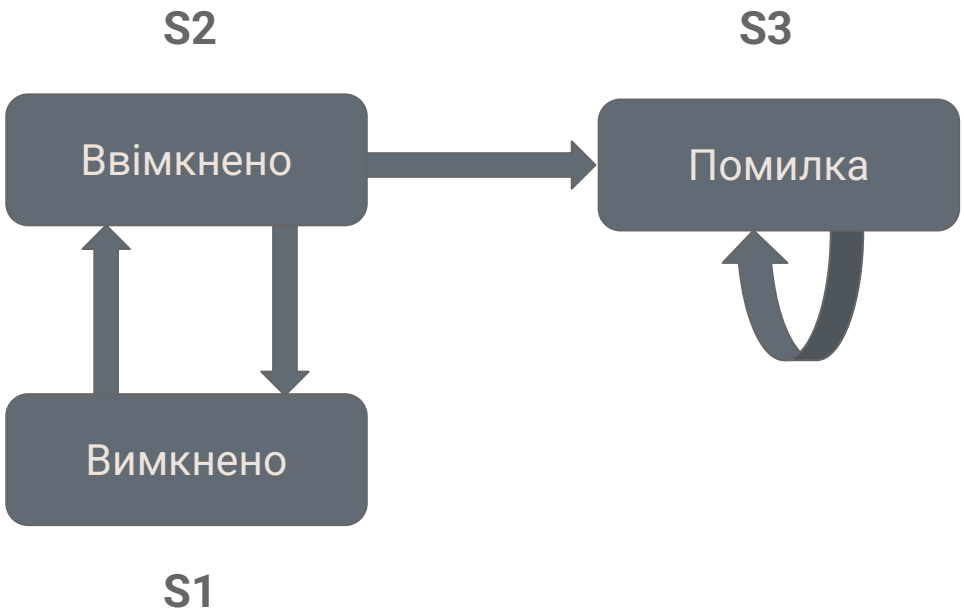
Грунтуючись на діаграмі переходу станів вмикача, який тест невалідний?

- Вимкнено-> ввімкнено
- Ввімкнено -> вимкнено
- **Помилка -> ввімкнено**
- Ввімкнено -> помилка

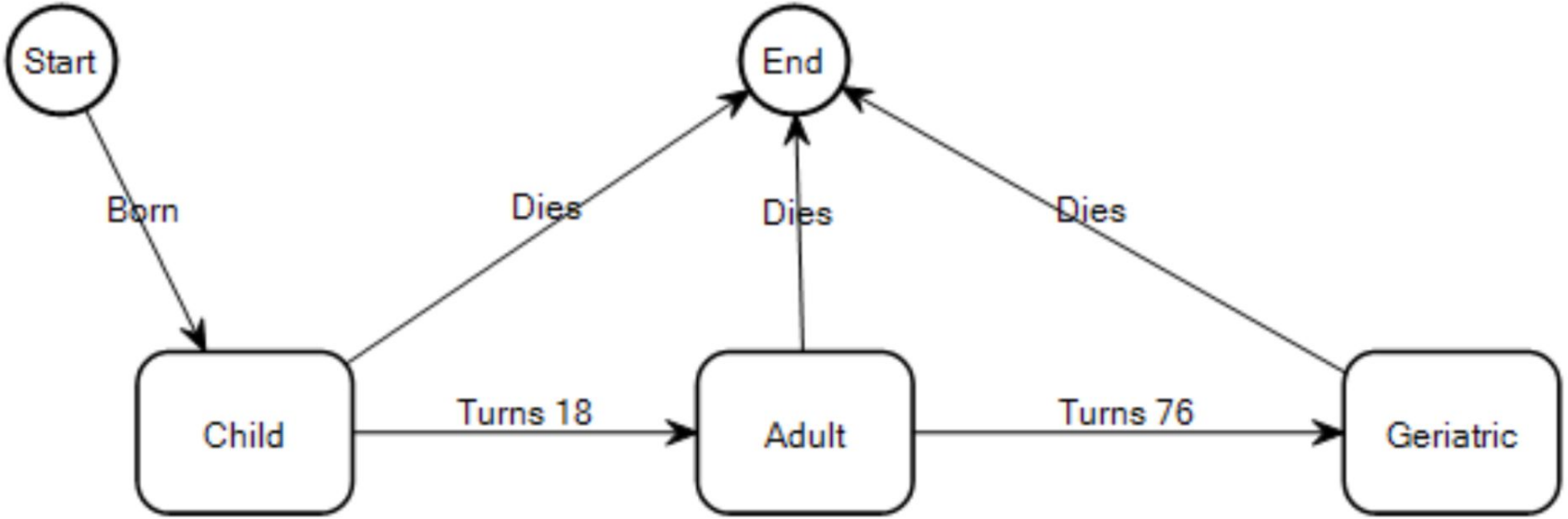


Грунтуючись на діаграмі переходу станів вмикача, який тест невалідний?

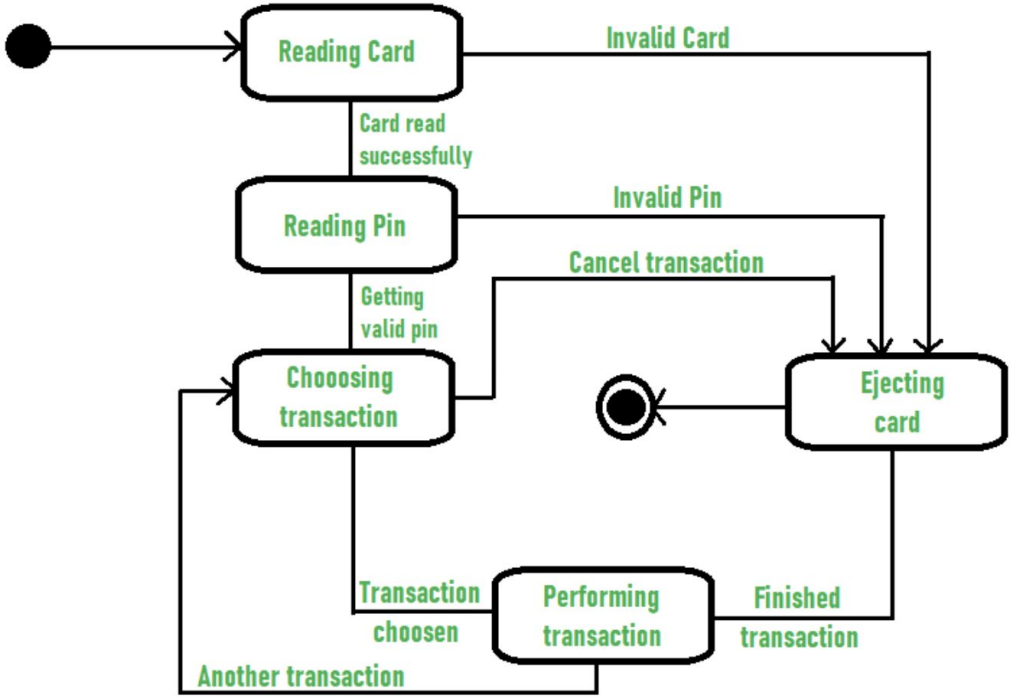
- Вимкнено-> ввімкнено
- Ввімкнено -> вимкнено
- **Помилка -> ввімкнено**
- Ввімкнено -> помилка



Приклад з відкритого доступу #1

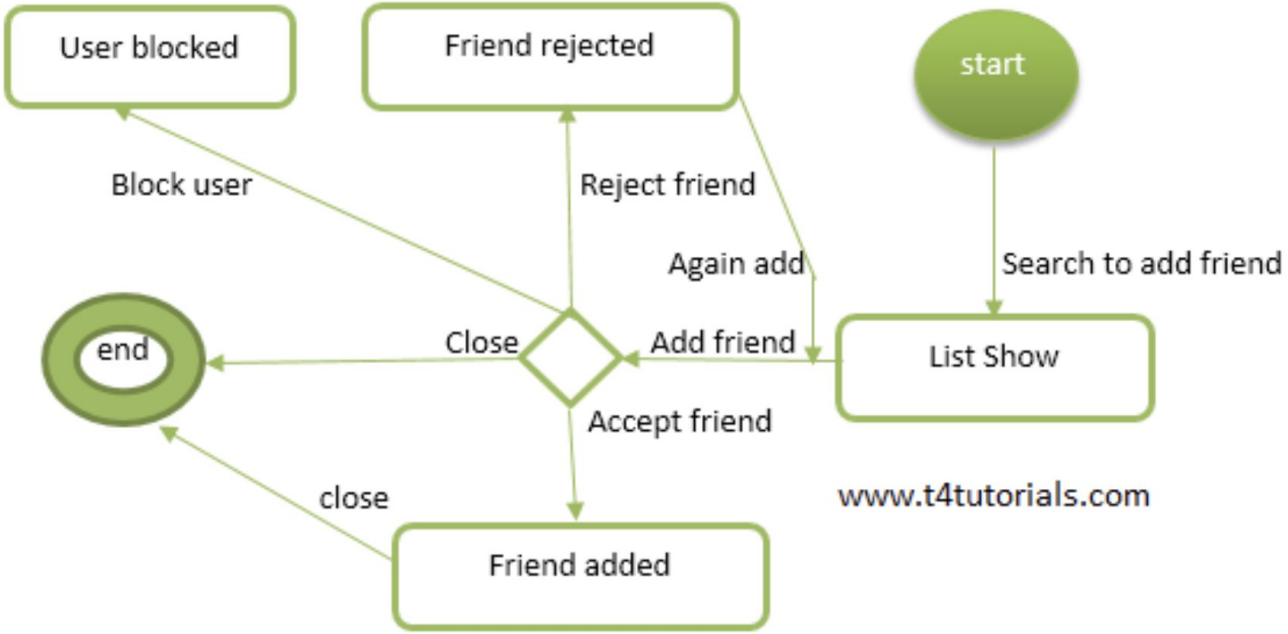


Приклад з відкритого доступу #2



State Transition Diagram for ATM System

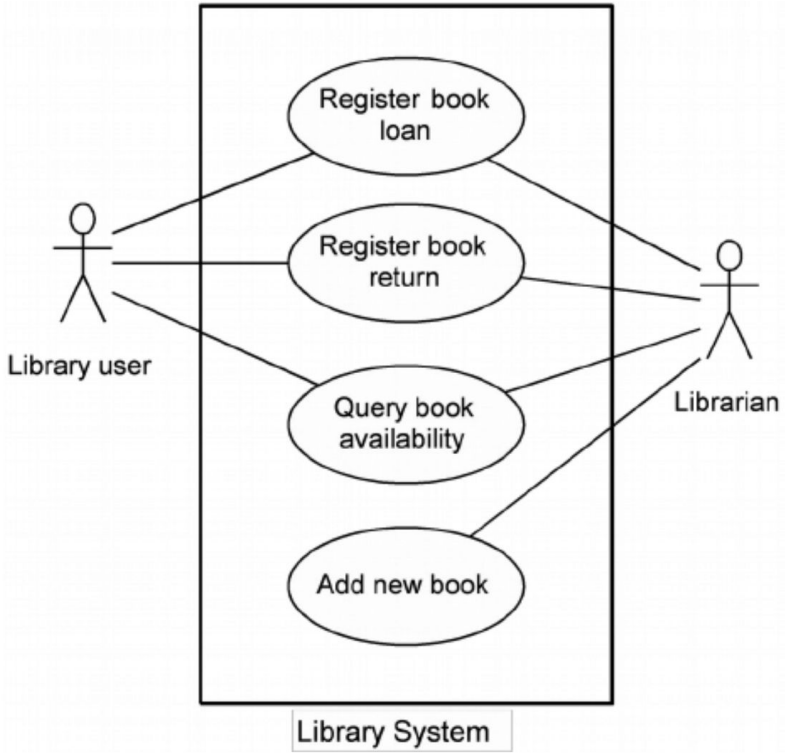
Приклад з відкритого доступу #3



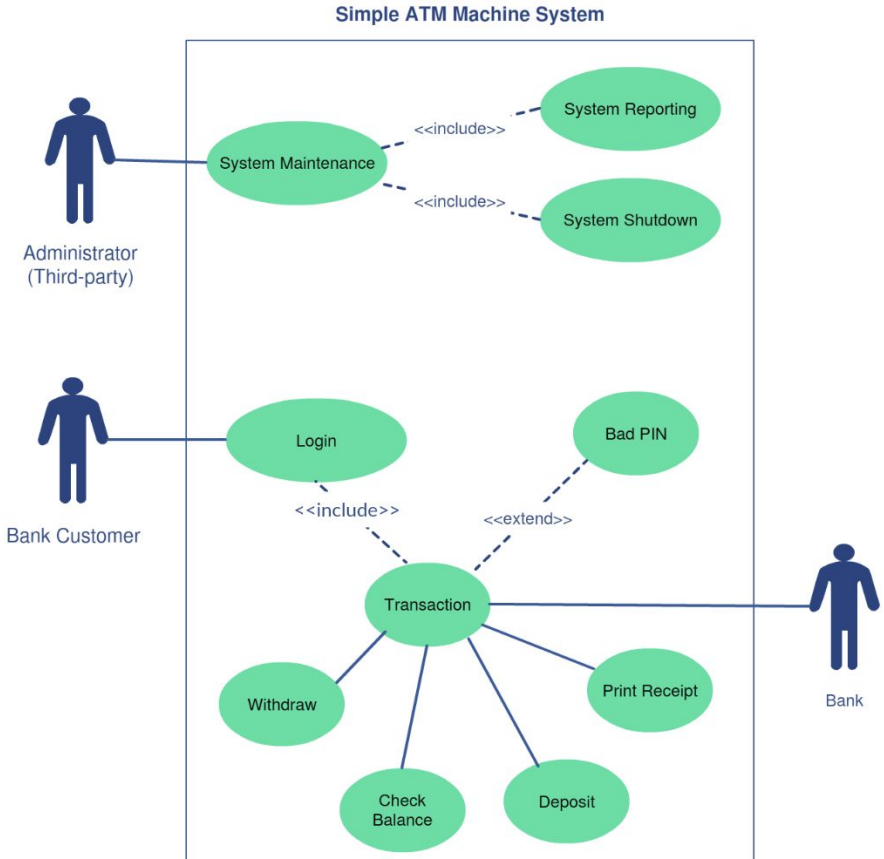
https://app.diagrams.net/

Діаграма сценаріїв використання

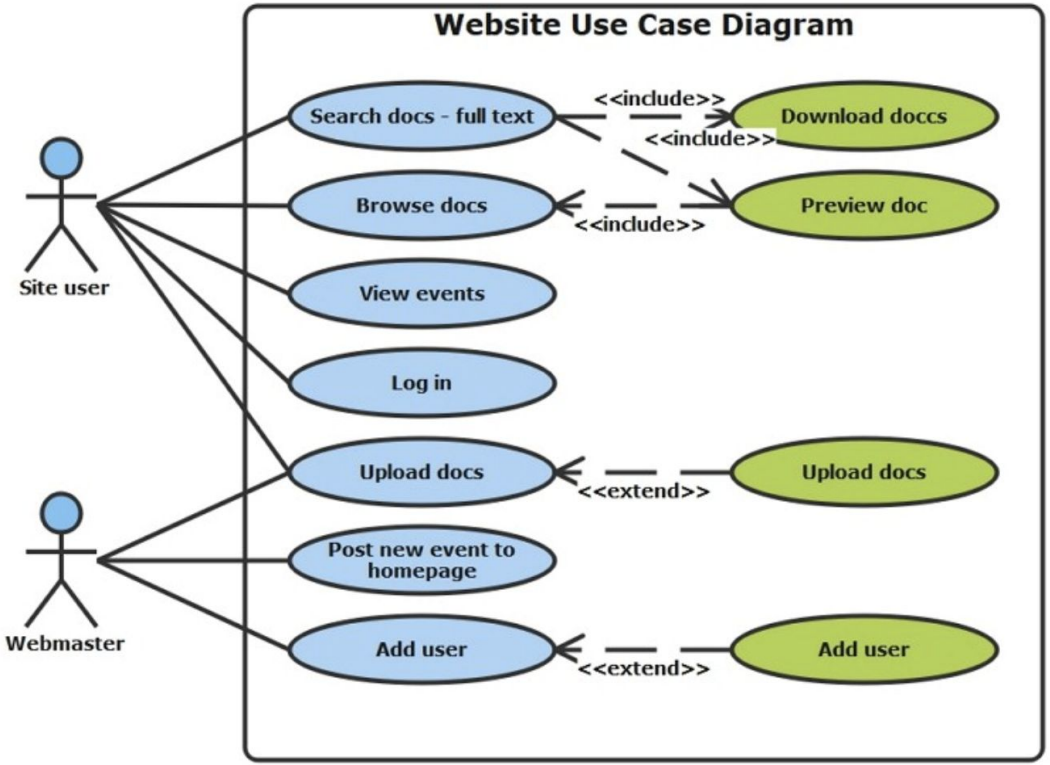
Приклад #1


Register book loan
Register book return
Query book availability
Add new book
Library user
Librarian
Library System

Приклад #2


Simple ATM Machine System
System Reporting
<<include>>
System Maintenance
<<include>>
System Shutdown
Administrator
(Third-party)
Login
Bad PIN
<<include>>
<<extend>>
Bank Customer
Transaction
Withdraw
Print Receipt
Check
Balance
Deposit
Bank

Приклад #3


Website Use Case Diagram
Search docs - full text
Download doccs
<<include>>
<<include>>
Browse docs
Preview doc
<<include>>
Site user
View events
Log in
Upload docs
Upload docs
<<extend>>
Post new event to homepage
Webmaster
Add user
Add user
<<extend>>

Тестування сценаріїв використання

- Сценарії використання описують взаємодію між користувачем (актором) та системою
- Це допомагає створювати тест кейси для інтеграційного, системного та приймального тестування.

Основний позитивний сценарій **A: актор** **C: система**	**Step**	**Опис**
	1	A: вставляє картку
	2	C: перевіряє картку і просить PIN
	3	A: вводить PIN
	4	C: перевіряє PIN
	5	C: дає доступ до аккаунту
Доповнення	2a	Карта невалідна. C: показати повідомлення і відхилити картку
	4a	PIN невалідний. C: показати повідомлення і попросити ввести PIN знову
	4б	PIN невалідний тричі C: З'їсти картку і вийти

Тестування сценаріїв використання

- Сценарії використання описують «потоки процесів» у системі з урахуванням її фактичного використання.
- Сценарії використання допомагають виявити дефекти у потоці процесу під час реального використання системи.
- Вони також допомагають виявити дефекти інтеграції, спричинені взаємодією різних компонентів.

https://app.diagrams.net/

Покриття операторів

Вступ до технік білого ящика

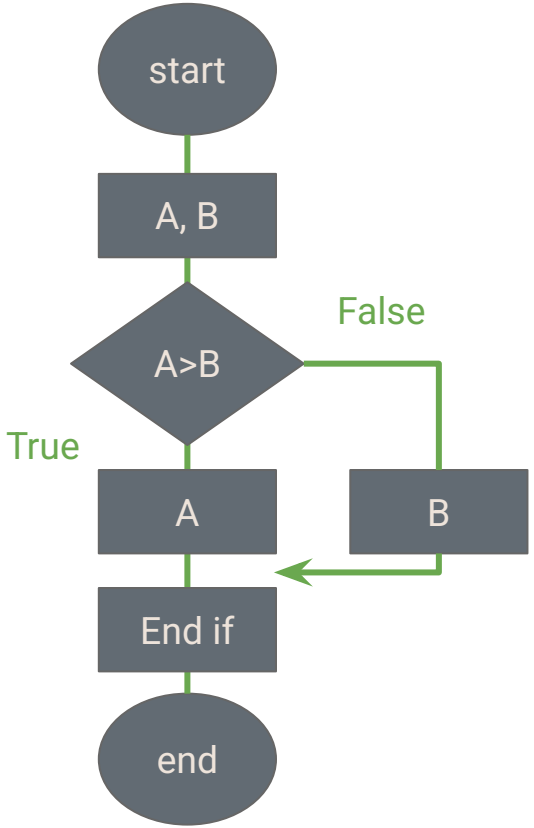
- Statement testing (тестування операторів) тестує оператори в даному фрагменті коду
- Decision testing (тестування умов) тестує умови в даному фрагменті коду
- Тестування умов ще відоме як Branch testing
- Тестування умов сильніше, ніж тестування операторів.
- 100% покриття умов гарантує 100% покриття операторів, але не навпаки.

Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
    Print “A більше”
Else
    Print “B більше”
End if
```

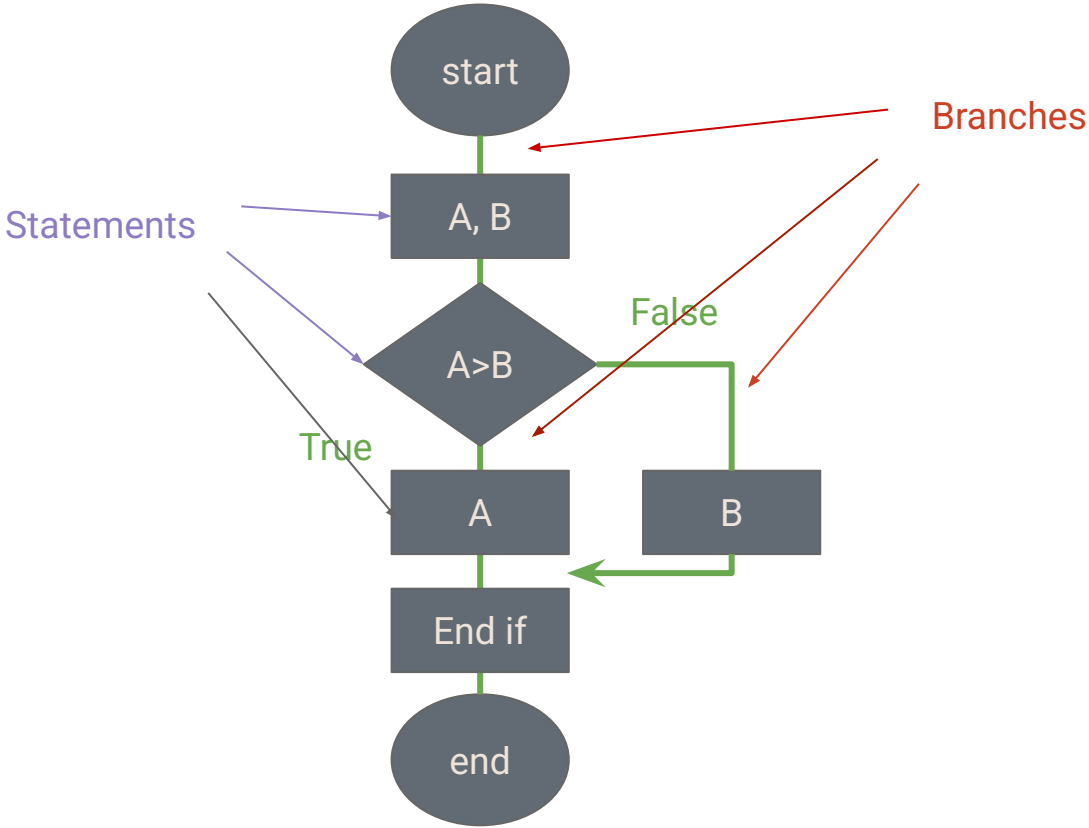
Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
    Print “A більше”
Else
    Print “B більше”
End if
```



Пояснювальна бригада - блок-схеми

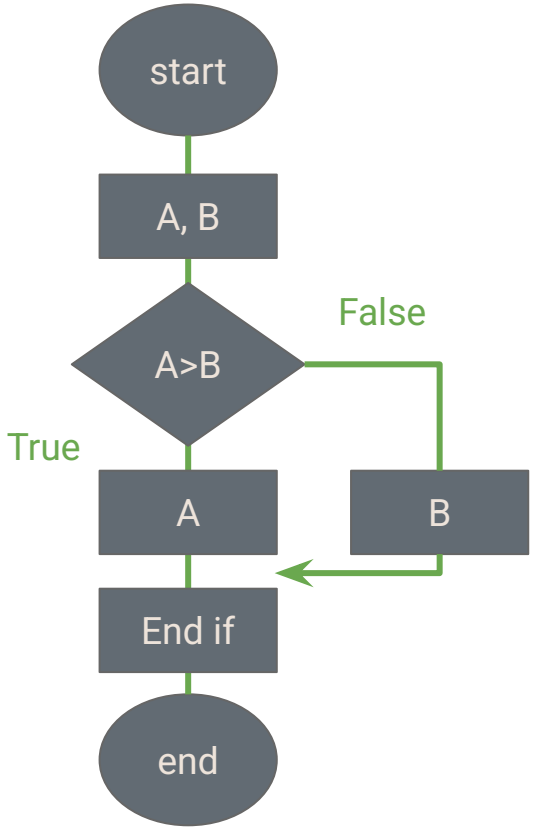
```
Read A
Read B
If A>B then
        Print “A більше”
Else
        Print “B більше”
End if
```



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

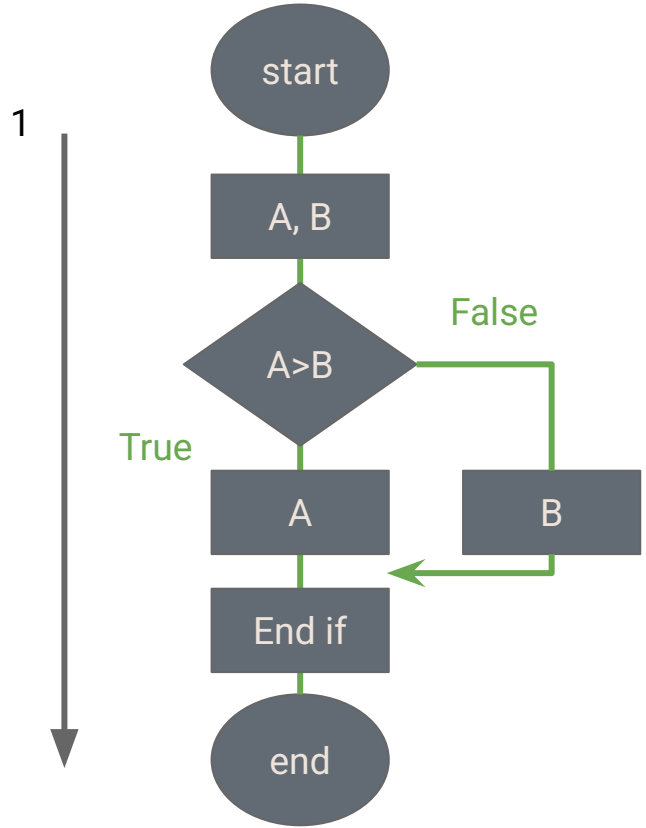
Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

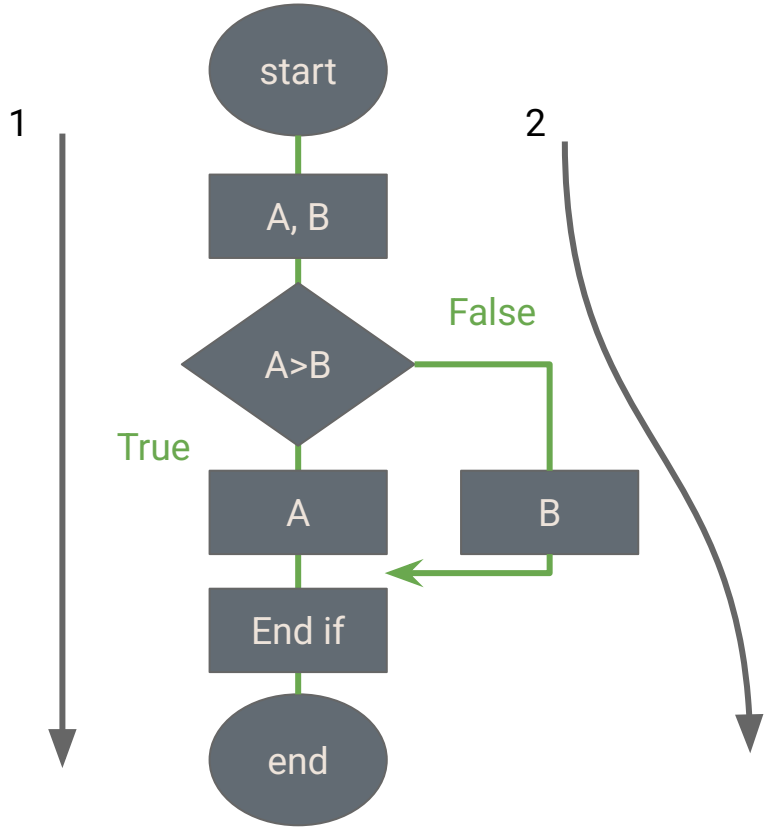
Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

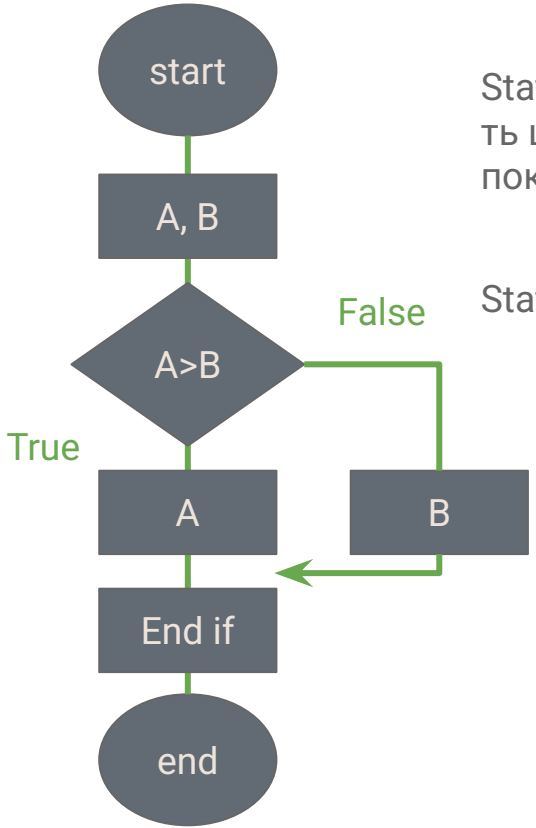
```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Скільки потрібно створити мінімум тест кейсів на 100% statement покриття?

Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Скільки потрібно створити мінімум тест кейсів на 100% statement покриття?



Statement coverage = мін. к-ть шляхів, потрібна для покриття всіх statements

Statement coverage = 2

Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

    показати “Введіть PIN-код”

    IF PIN валідний THEN

        вибрати транзакцію

    ELSE

        показати “PIN невалідний”

ELSE

відхилити карту

Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

показати “Введіть PIN-код”

IF PIN валідний THEN

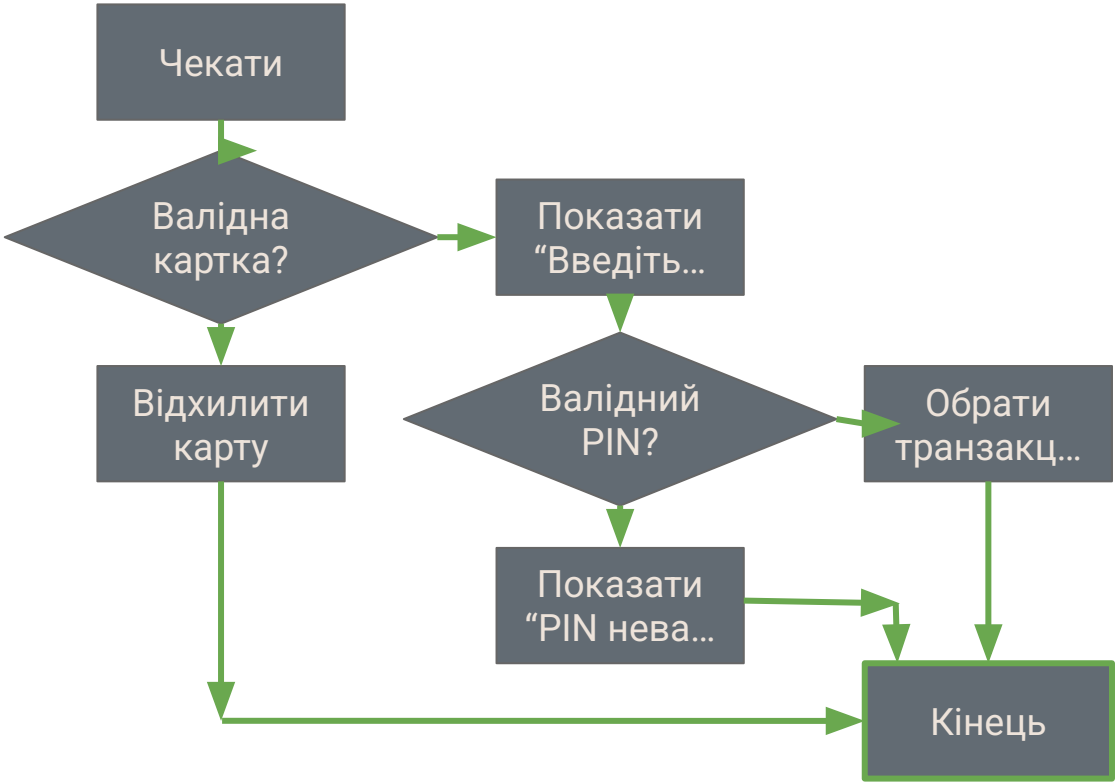
вибрати транзакцію

ELSE

показати “PIN невалідний”

ELSE

відхилити карту



Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

показати “Введіть PIN-код”

IF PIN валідний THEN

вибрати транзакцію

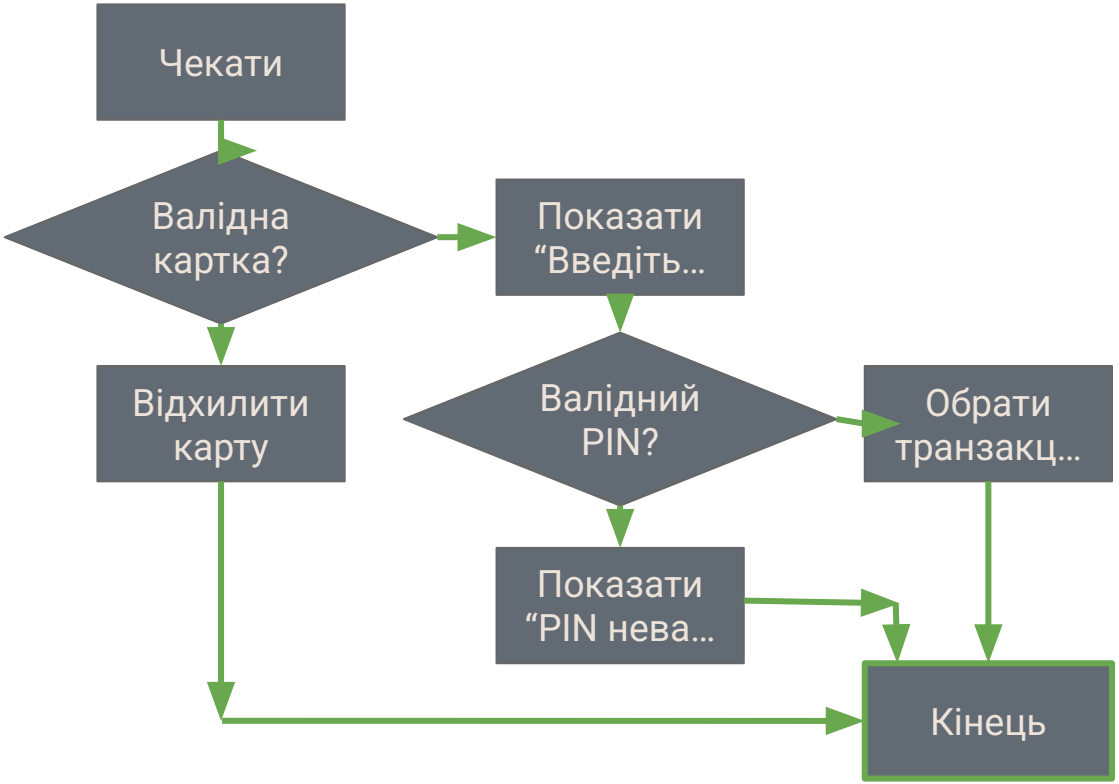
ELSE

показати “PIN невалідний”

ELSE

відхилити карту

Statement coverage = 3



Приклад #1

TIP: Statement coverage = 1 + к-ть ELSE

Чекати, щоб вставили картку

IF карта валідна THEN

    показати “Введіть PIN-код”

    IF PIN валідний THEN

        вибрати транзакцію

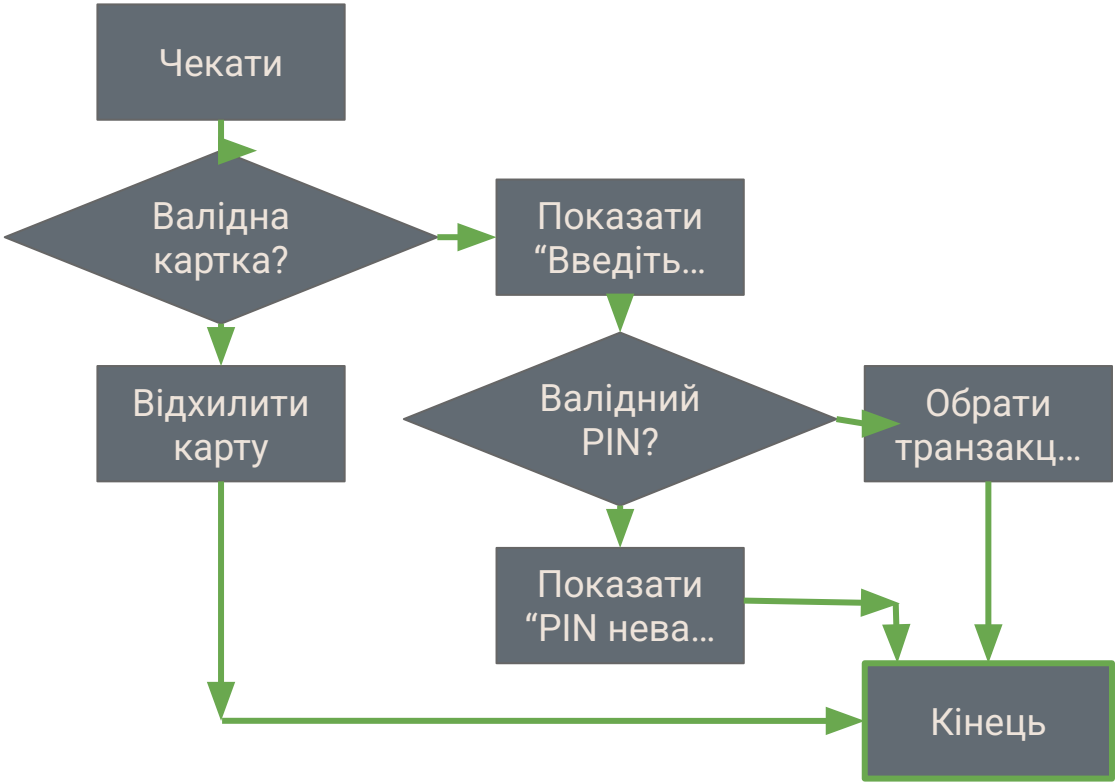
    ELSE

        показати “PIN невалідний”

ELSE

відхилити карту

Statement coverage = 3

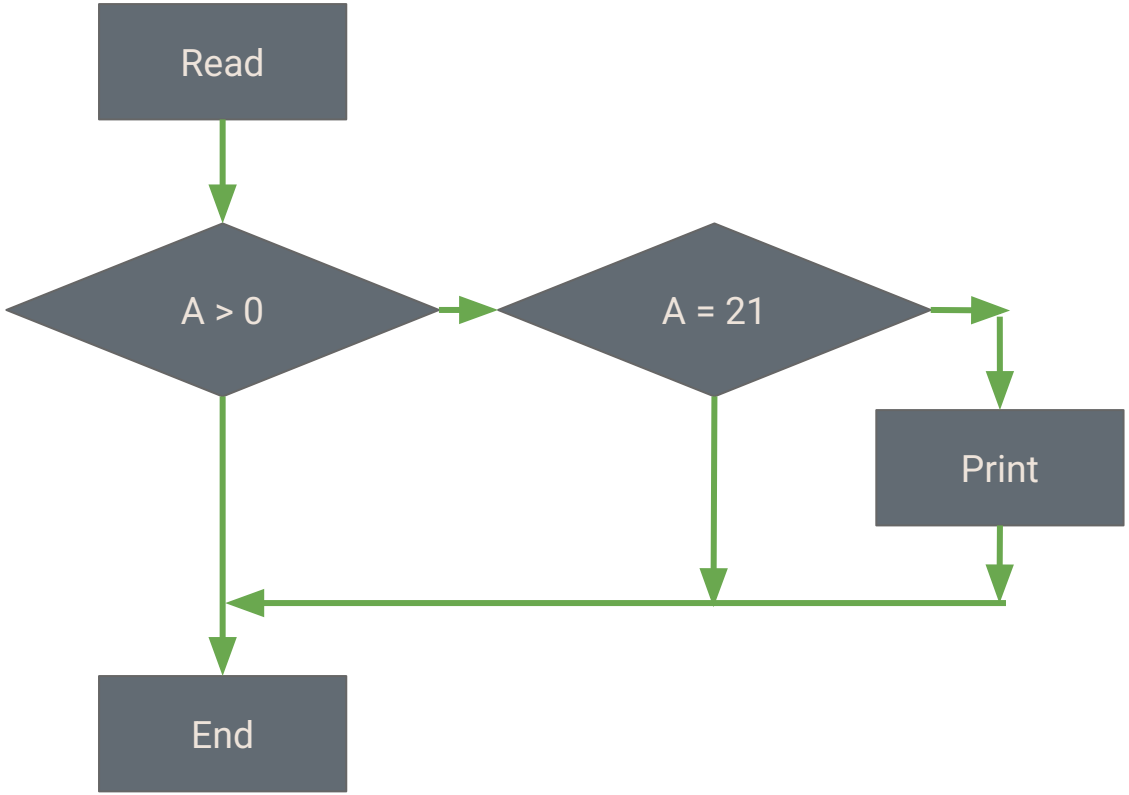


Приклад #2

```
Read A
IF A > 0 THEN
        IF A = 21 THEN
        Print “Key”
        ENDIF
ENDIF
```

Приклад #2

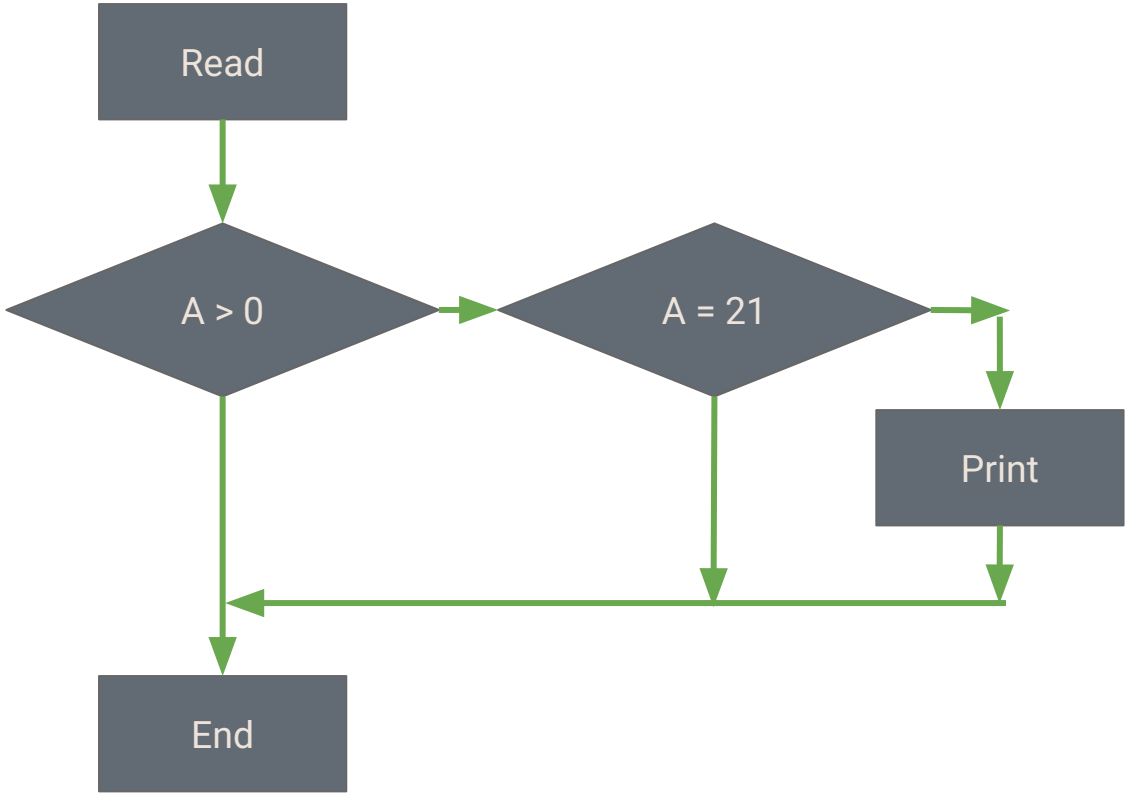
```
Read A
IF A > 0 THEN
        IF A = 21 THEN
        Print “Key”
        ENDIF
ENDIF
```



Приклад #2

```
Read A
IF A > 0 THEN
    IF A = 21 THEN
    Print “Key”
    ENDIF
ENDIF
```

Statement coverage = 1

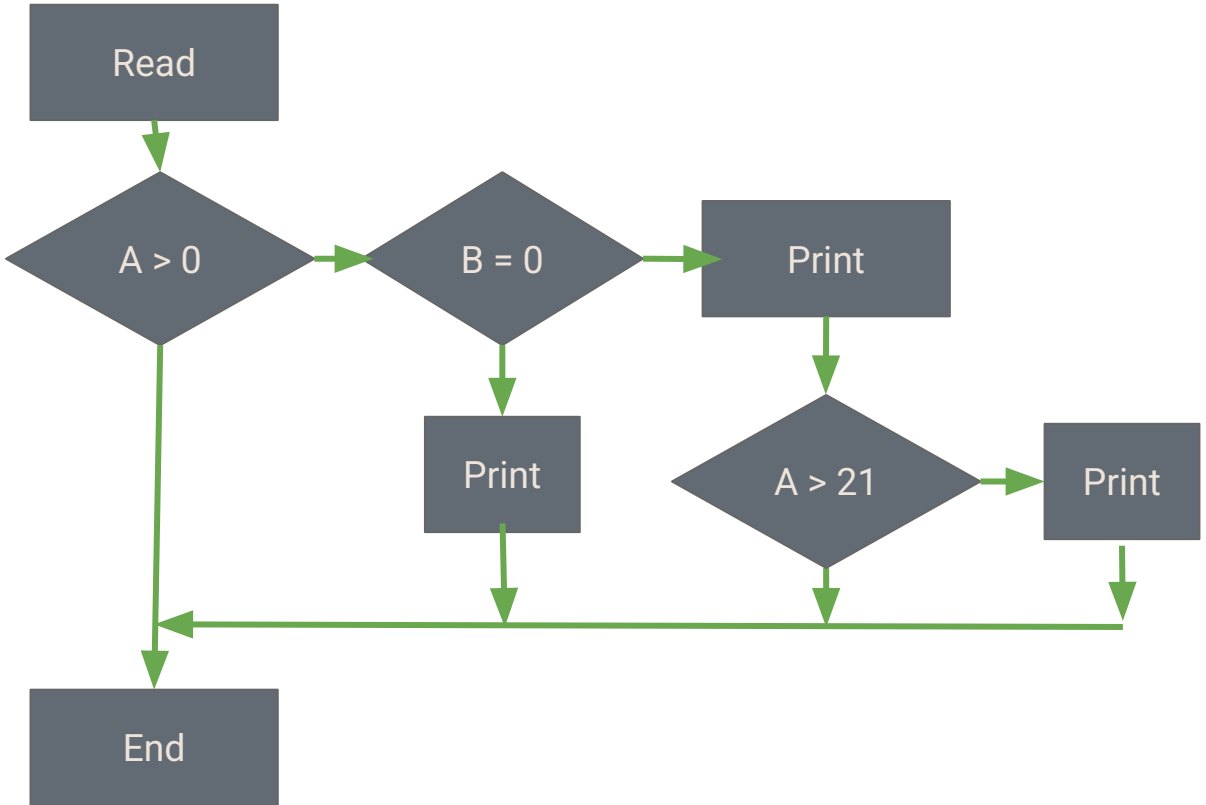


Приклад #3

```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
                IF A > 21 THEN
                        Print A
                ENDIF
        ENDIF
ENDIF
```

Приклад #3

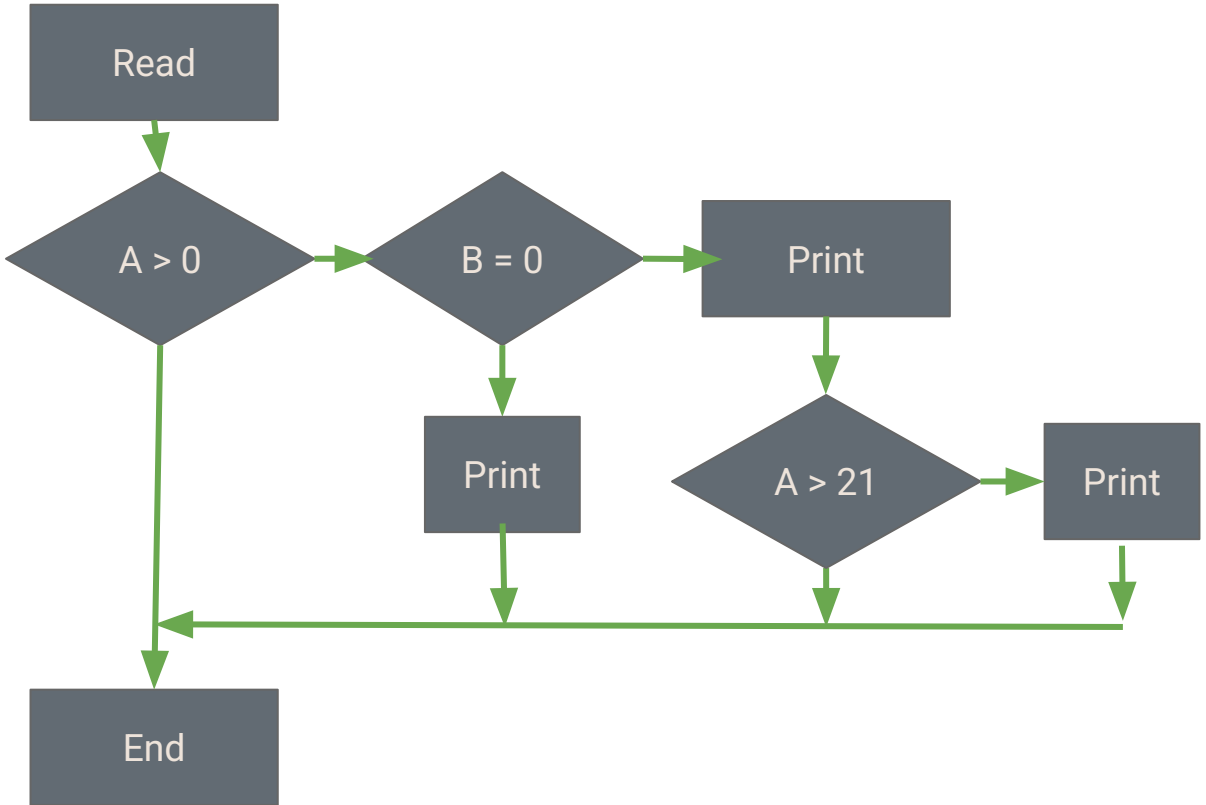
```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
                IF A > 21 THEN
                        Print A
                ENDIF
        ENDIF
ENDIF
```



Приклад #3

```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
                IF A > 21 THEN
                        Print A
                ENDIF
        ENDIF
ENDIF
```

Statement coverage = 2



Приклад #4

```
Read A
Read B
IF A < 0 THEN
        Print “A negative”
ELSE
        Print “A positive”
ENDIF
IF B < 0 THEN
        Print “B negative”
ELSE
Print “B positive”
ENDIF
```

Приклад #4

Read A

Read B

IF A < 0 THEN

    Print “A negative”

ELSE

    Print “A positive”

ENDIF

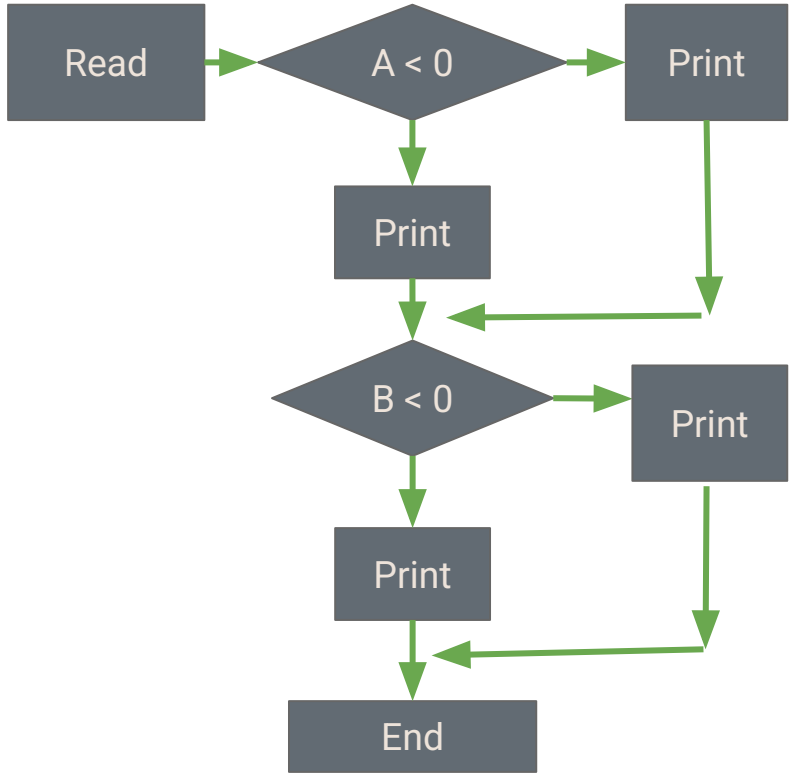
IF B < 0 THEN

    Print “B negative”

ELSE

Print “B positive”

ENDIF



Приклад #4

Read A

Read B

IF A < 0 THEN

    Print “A negative”

ELSE

    Print “A positive”

ENDIF

IF B < 0 THEN

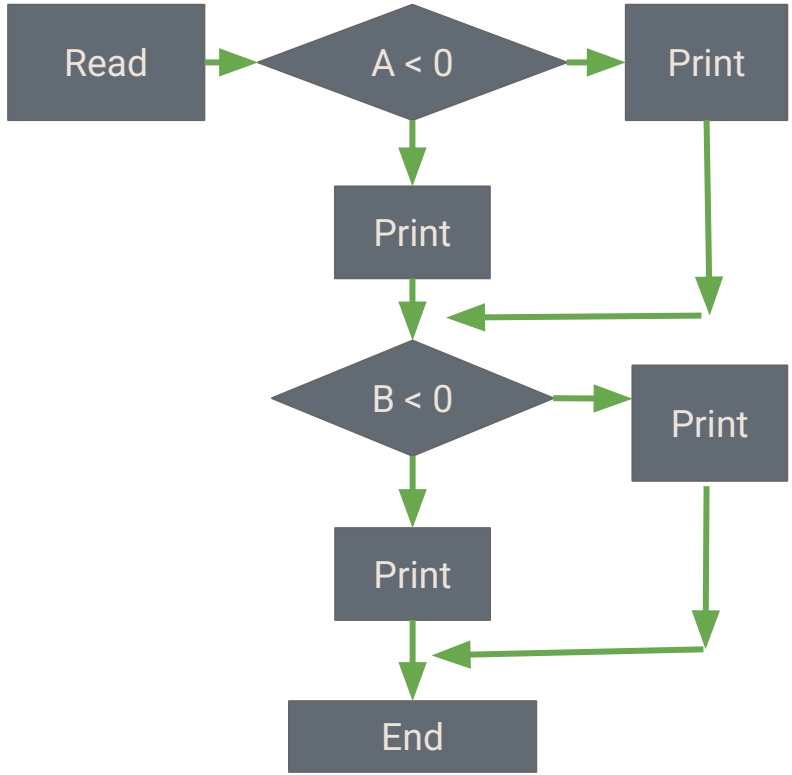
    Print “B negative”

ELSE

Print “B positive”

ENDIF

Statement coverage = 2



Приклад #4

Read A

Read B

IF A < 0 THEN

    Print “A negative”

ELSE

    Print “A positive”

ENDIF

IF B < 0 THEN

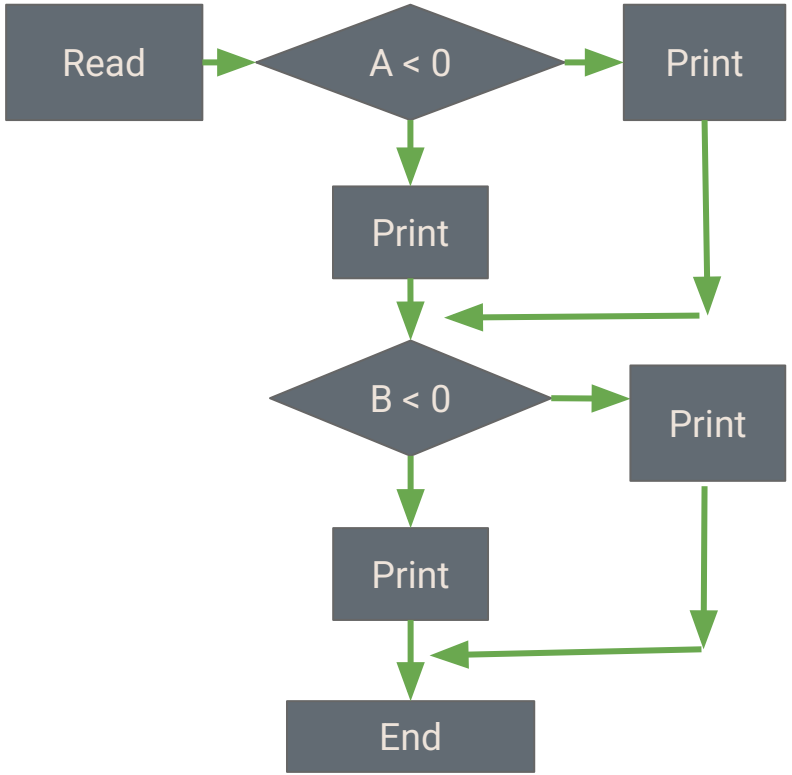
    Print “B negative”

ELSE

Print “B positive”

ENDIF

Statement coverage = 2



TIP: Якщо є ELSE Statement покриття = 2

Якщо немає ELSE Statement покриття = 1

Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке statement покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 75%
3. 90%
4. 100%

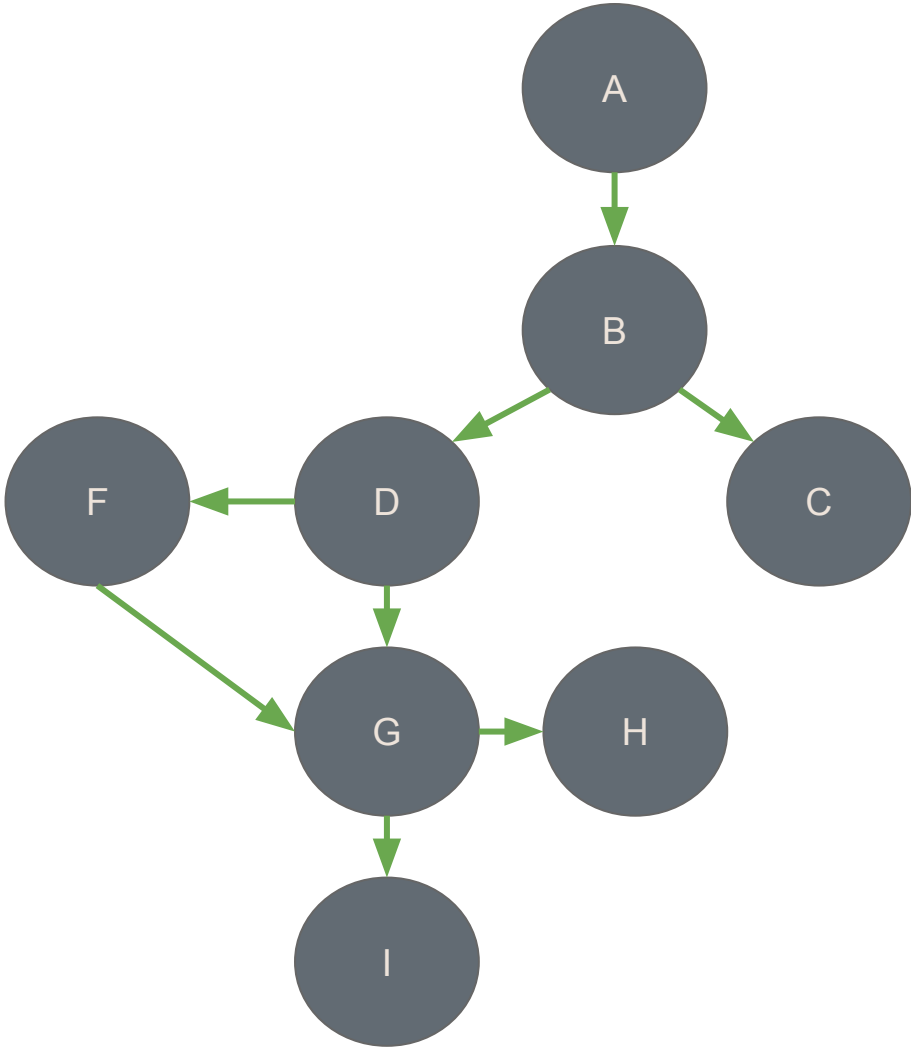
Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке statement покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 75%
3. 90%
4. 100%



Приклад #5

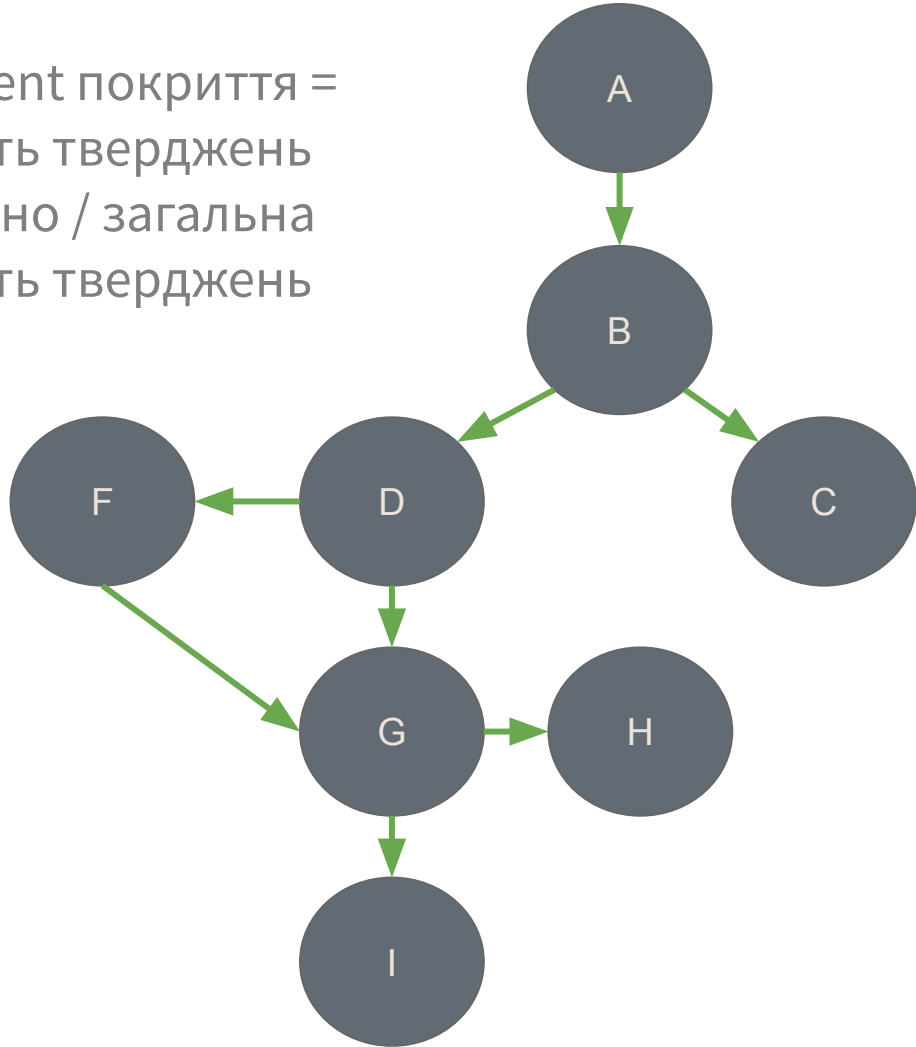
Для цього фрагмента коду було проведено такі шляхи/тести. Яке statement покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 75%
3. 90%
4. 100%

Statement покриття = кількість тверджень виконано / загальна кількість тверджень



Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке statement покриття досягнуто?

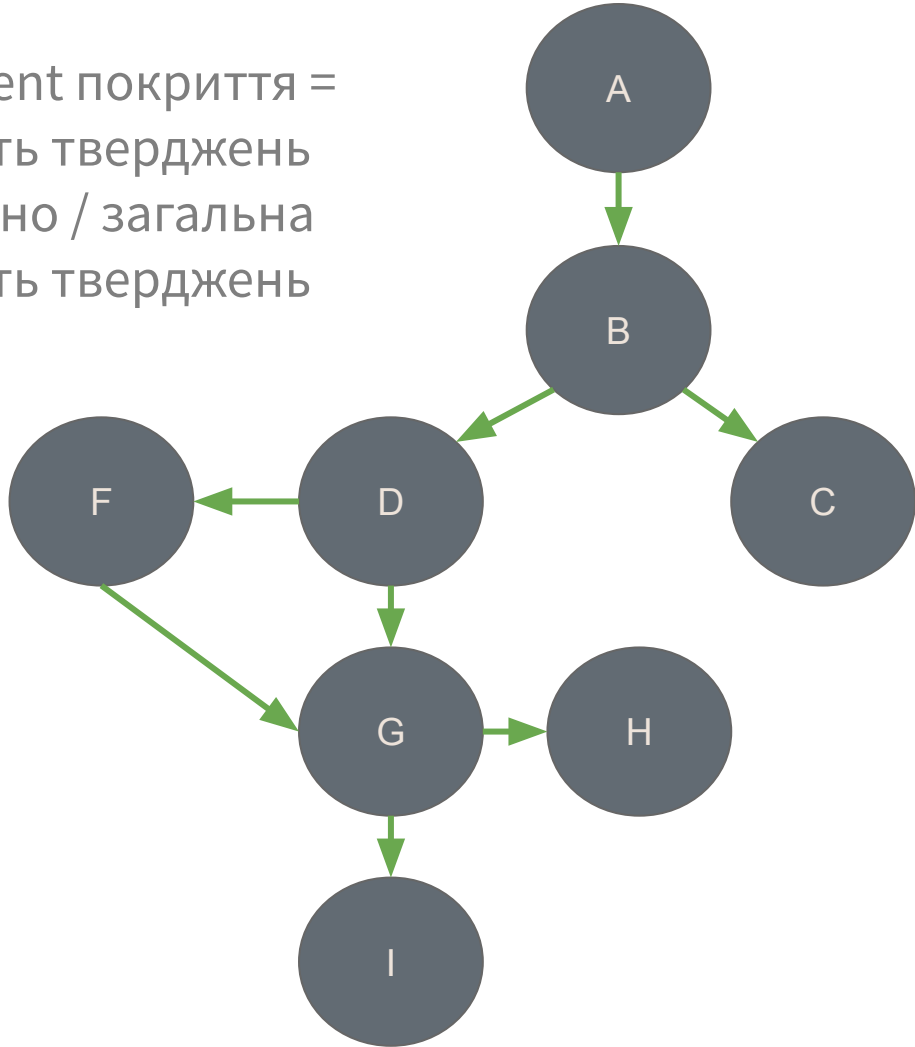
Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 75%
3. 90%
4. 100%

Statement покриття = кількість тверджень виконано / загальна кількість тверджень

6 із 8 тверджень перевірено



Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке statement покриття досягнуто?

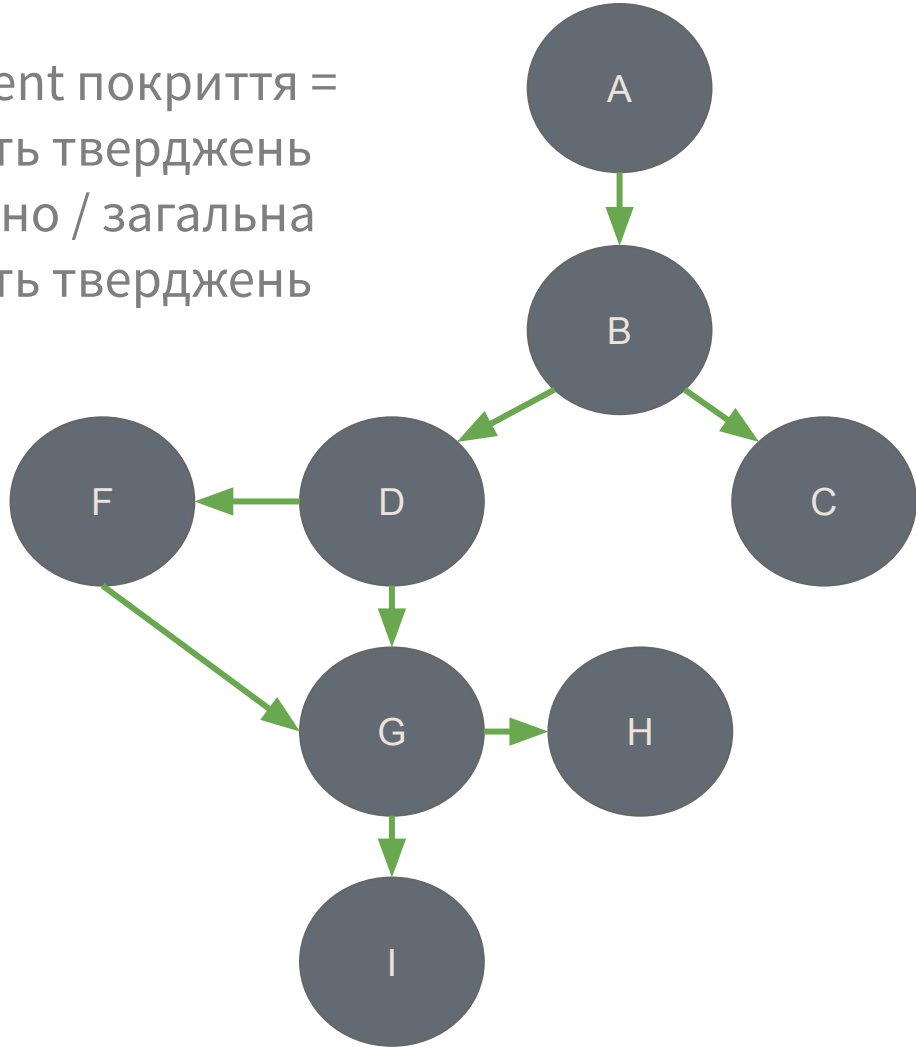
Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. **75%**
3. 90%
4. 100%

6 із 8 тверджень перевірено

Statement покриття = кількість тверджень виконано / загальна кількість тверджень



Покриття умов

Вступ до технік білого ящика

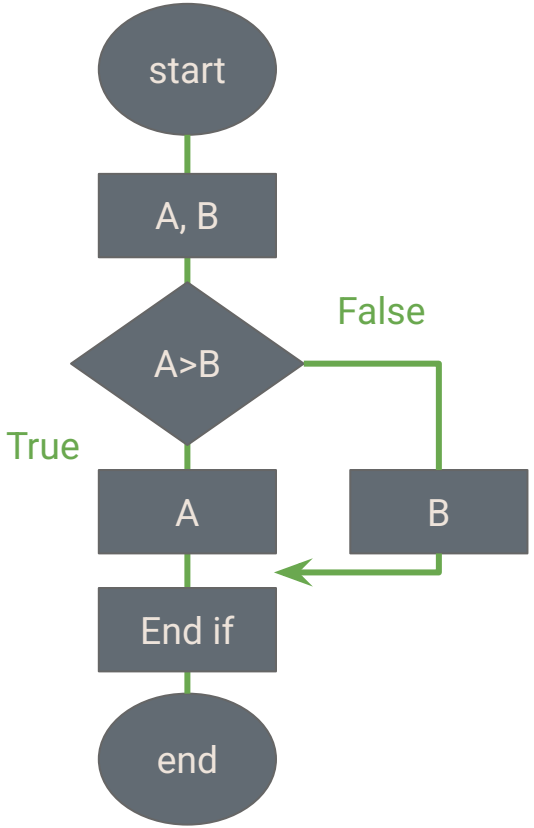
- Statement testing (тестування операторів) тестує оператори в даному фрагменті коду
- Decision testing (тестування умов) тестує умови в даному фрагменті коду
- Тестування умов ще відоме як Branch testing
- Тестування умов сильніше, ніж тестування операторів.
- 100% покриття умов гарантує 100% покриття операторів, але не навпаки.

Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
    Print “A більше”
Else
    Print “B більше”
End if
```

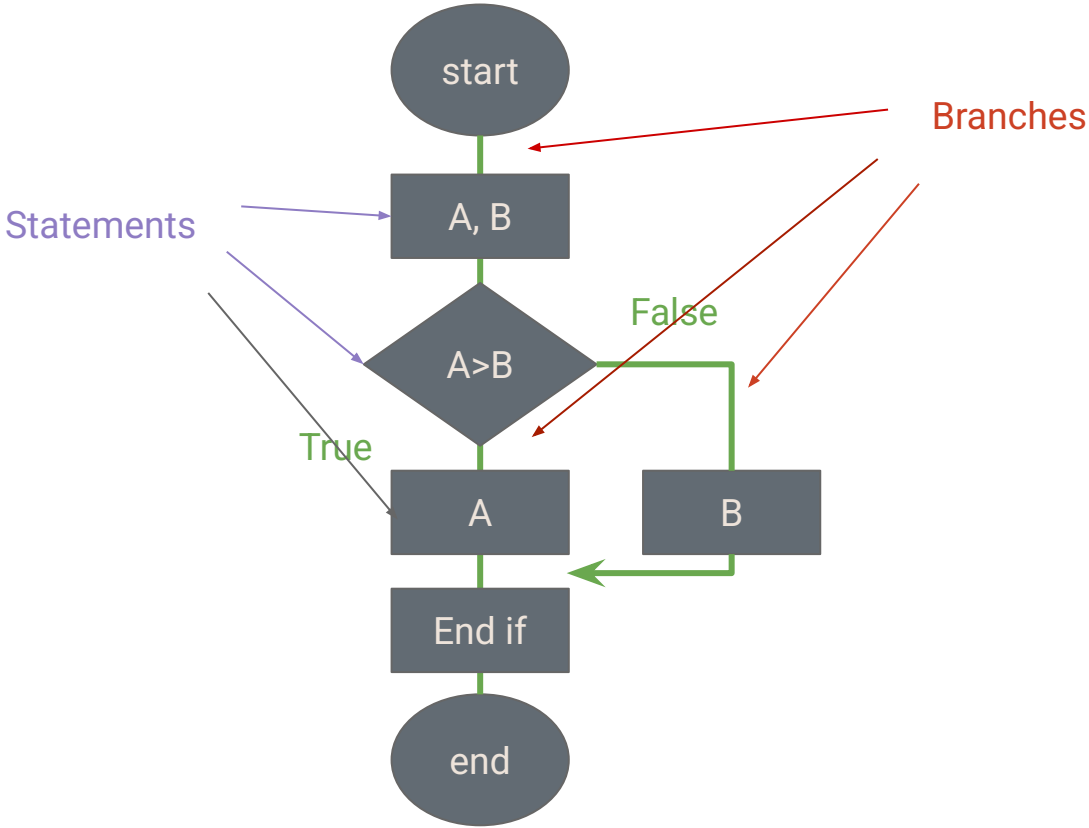
Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
    Print “A більше”
Else
    Print “B більше”
End if
```



Пояснювальна бригада - блок-схеми

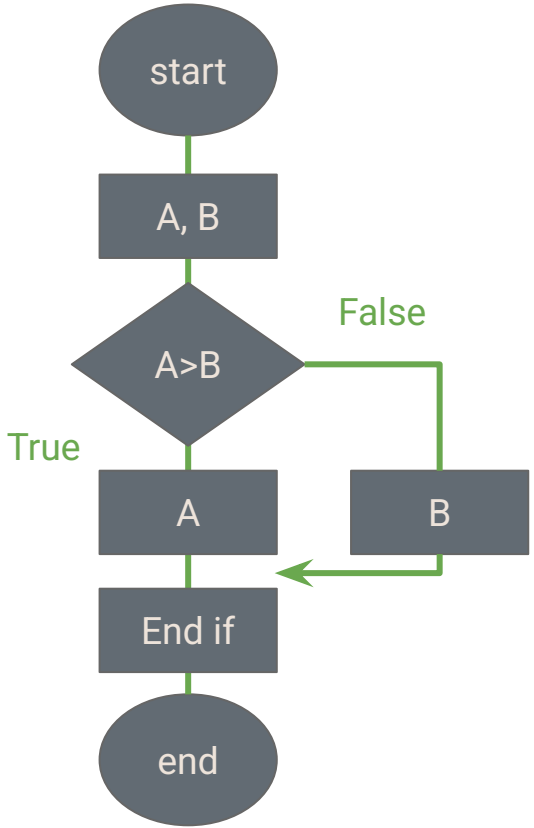
```
Read A
Read B
If A>B then
        Print “A більше”
Else
        Print “B більше”
End if
```



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

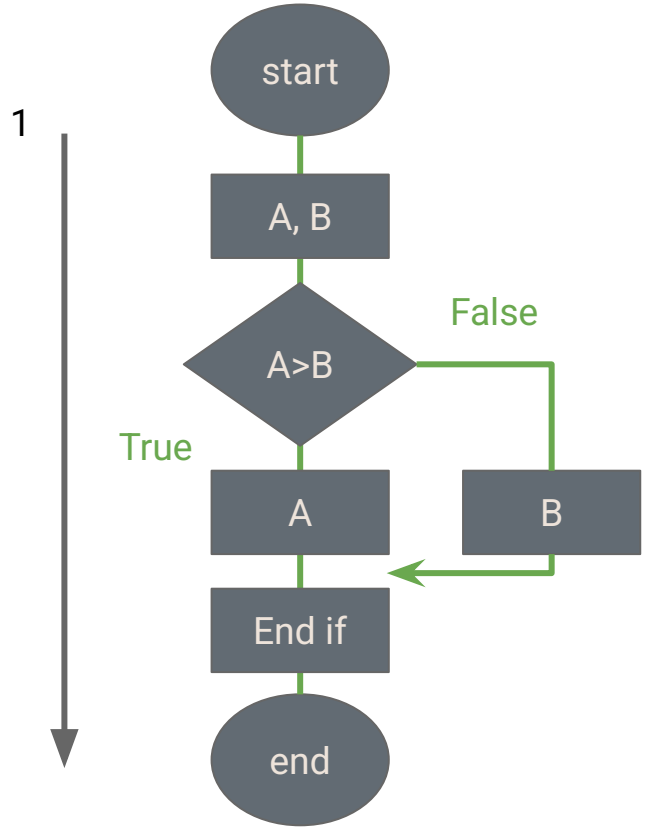
Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

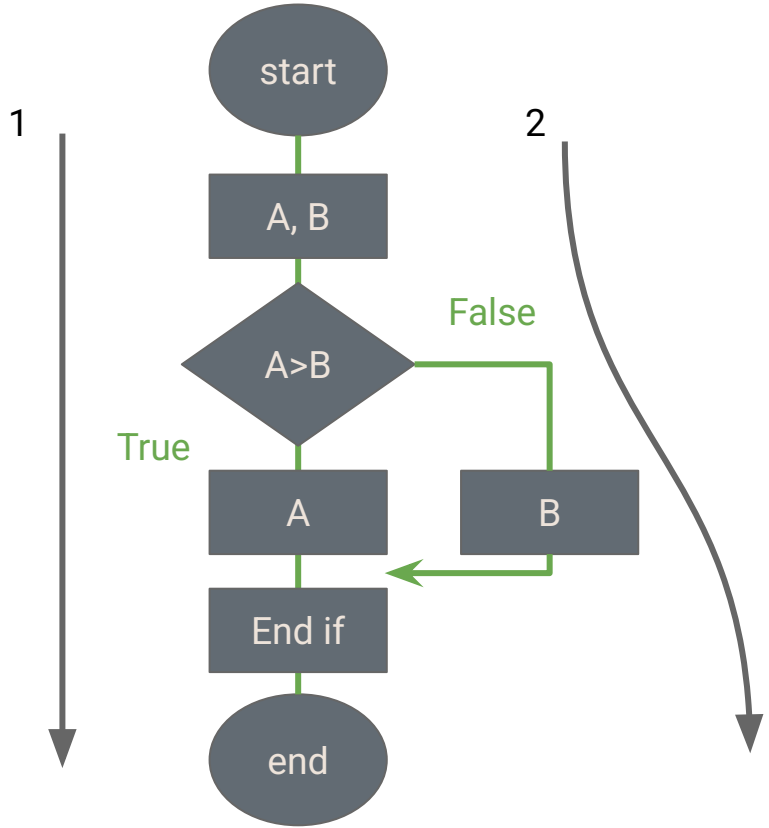
Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Шлях - це проходження по коду, починаючи з точки старт і закінчуючи точкою кінець.



Пояснювальна бригада - блок-схеми

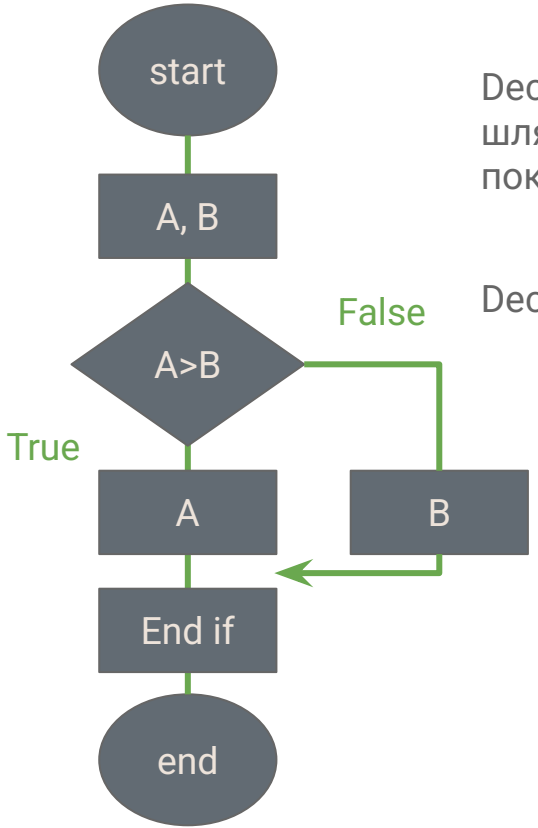
```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Скільки потрібно створити мінімум тест кейсів на 100% decision покриття?

Пояснювальна бригада - блок-схеми

```
Read A
Read B
If A>B then
      Print “A більше”
Else
      Print “B більше”
End if
```

Скільки потрібно створити мінімум тест кейсів на 100% decision покриття?



Decision coverage = мін. к-ть шляхів, потрібна для покриття всіх branches

Decision coverage = 2

Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

    показати “Введіть PIN-код”

    IF PIN валідний THEN

        вибрати транзакцію

    ELSE

        показати “PIN невалідний”

ELSE

відхилити карту

Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

показати “Введіть PIN-код”

IF PIN валідний THEN

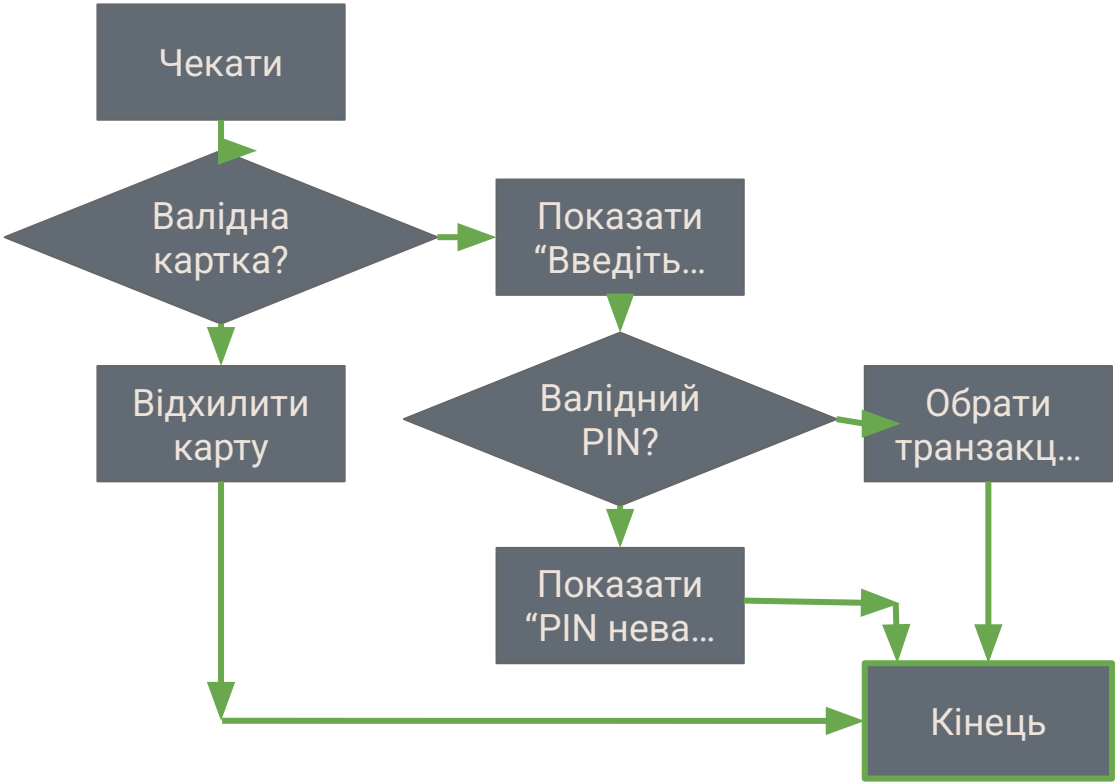
вибрати транзакцію

ELSE

показати “PIN невалідний”

ELSE

відхилити карту



Приклад #1

Чекати, щоб вставили картку

IF карта валідна THEN

показати “Введіть PIN-код”

IF PIN валідний THEN

вибрати транзакцію

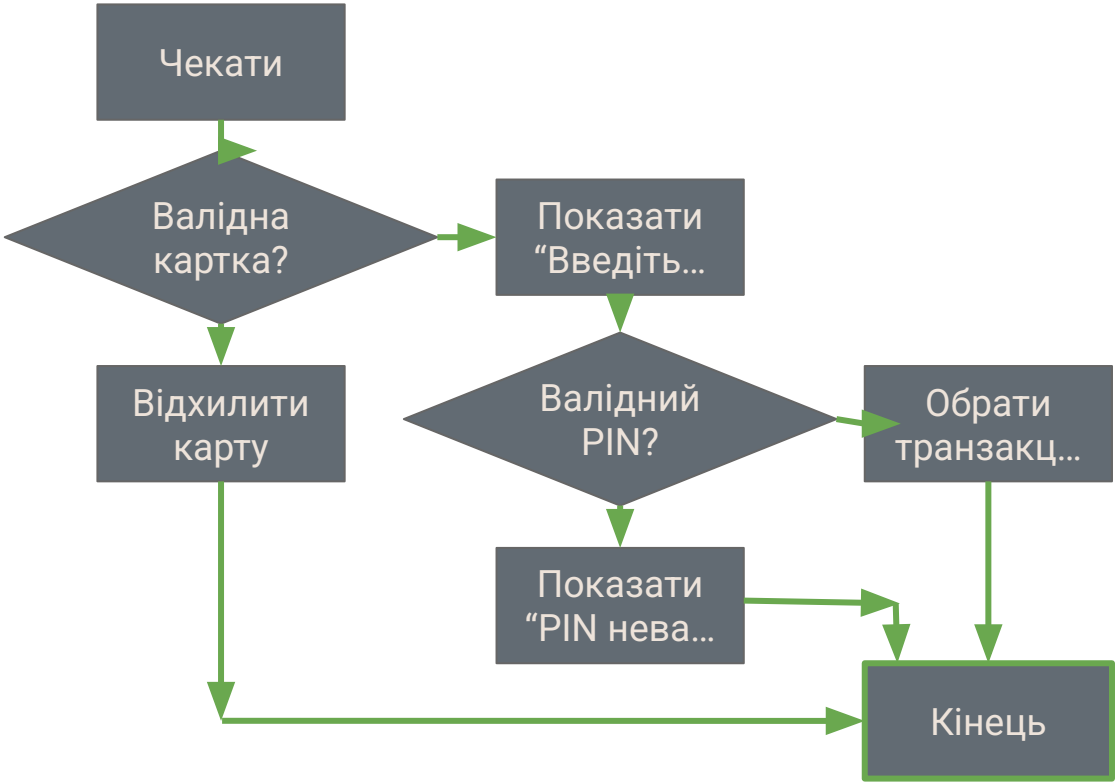
ELSE

показати “PIN невалідний”

ELSE

відхилити карту

Decision coverage = 3



Приклад #1

TIP: Decision coverage = 1 + к-ть IF (у statement - else)

Чекати, щоб вставили картку

IF карта валідна THEN

показати “Введіть PIN-код”

IF PIN валідний THEN

вибрати транзакцію

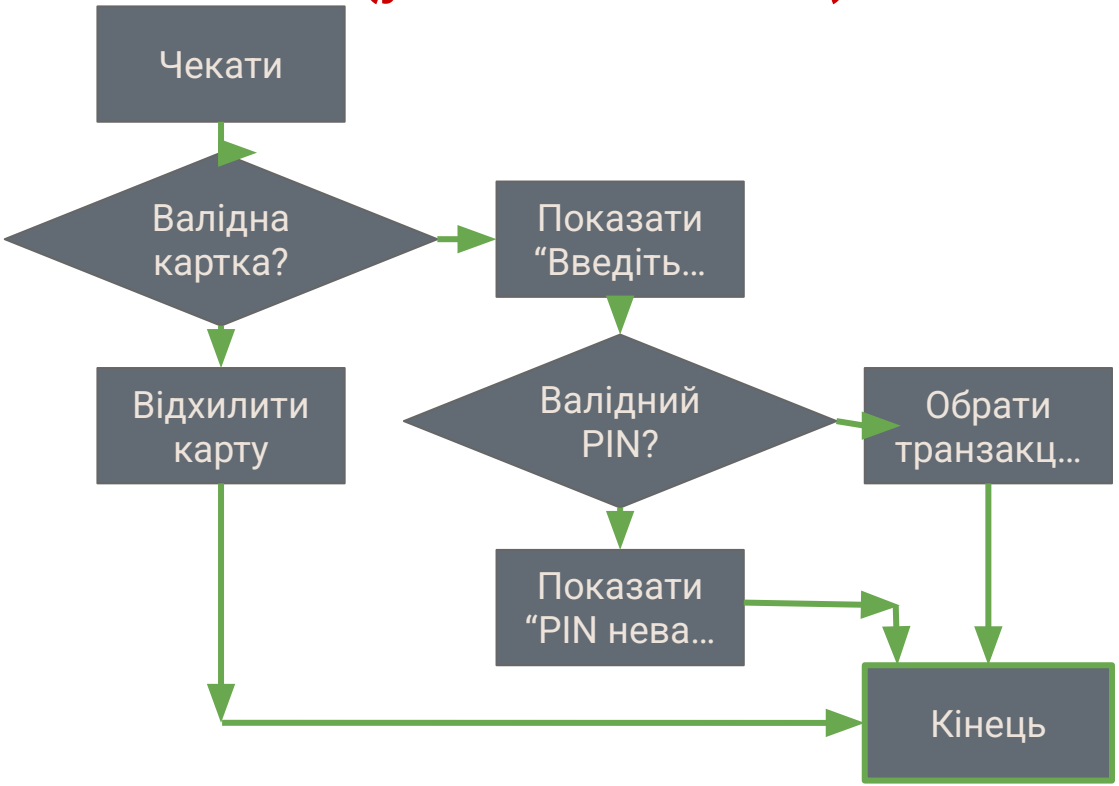
ELSE

показати “PIN невалідний”

ELSE

відхилити карту

Decision coverage = 3

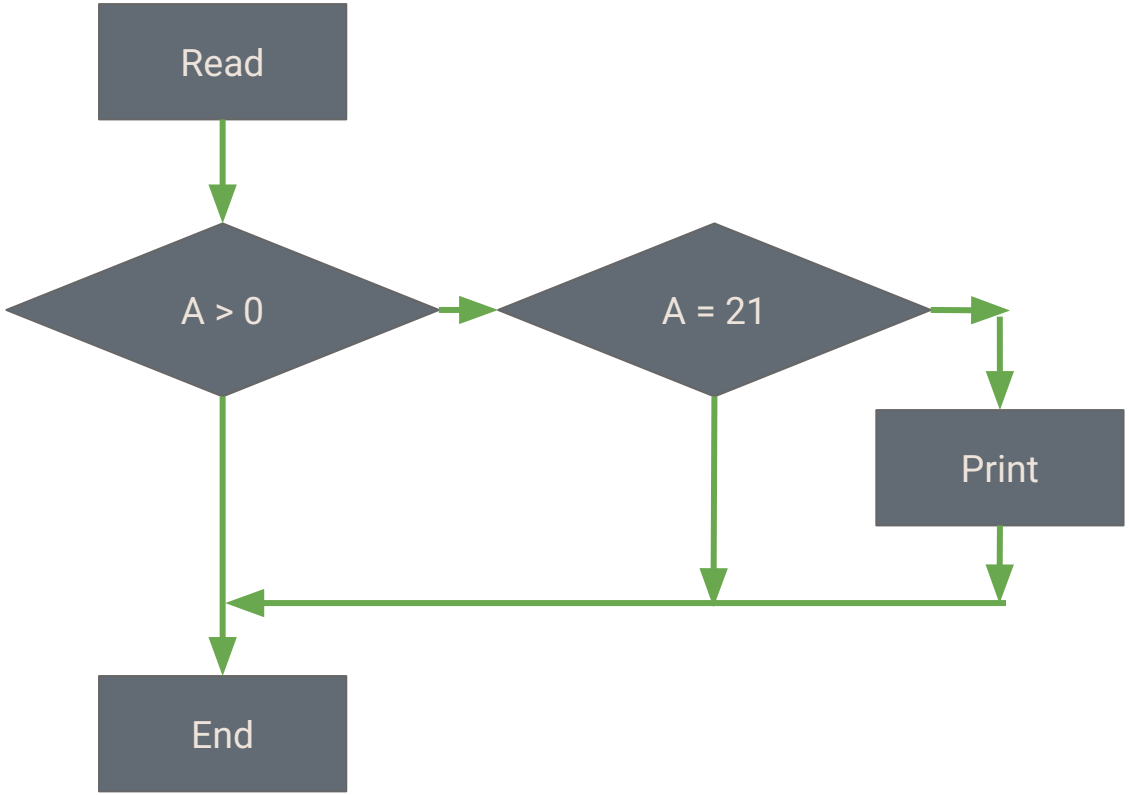


Приклад #2

```
Read A
IF A > 0 THEN
        IF A = 21 THEN
        Print “Key”
        ENDIF
ENDIF
```

Приклад #2

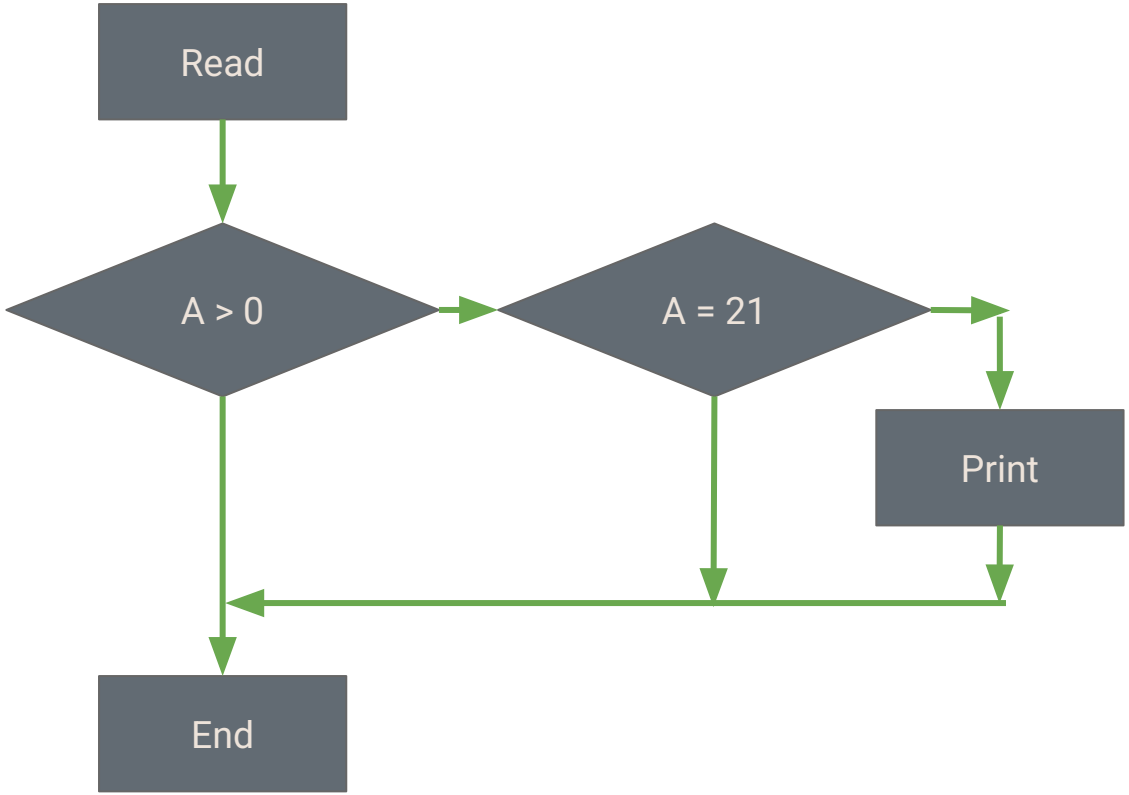
```
Read A
IF A > 0 THEN
        IF A = 21 THEN
        Print “Key”
        ENDIF
ENDIF
```



Приклад #2

```
Read A
IF A > 0 THEN
        IF A = 21 THEN
        Print “Key”
        ENDIF
ENDIF
```

Decision coverage = 3

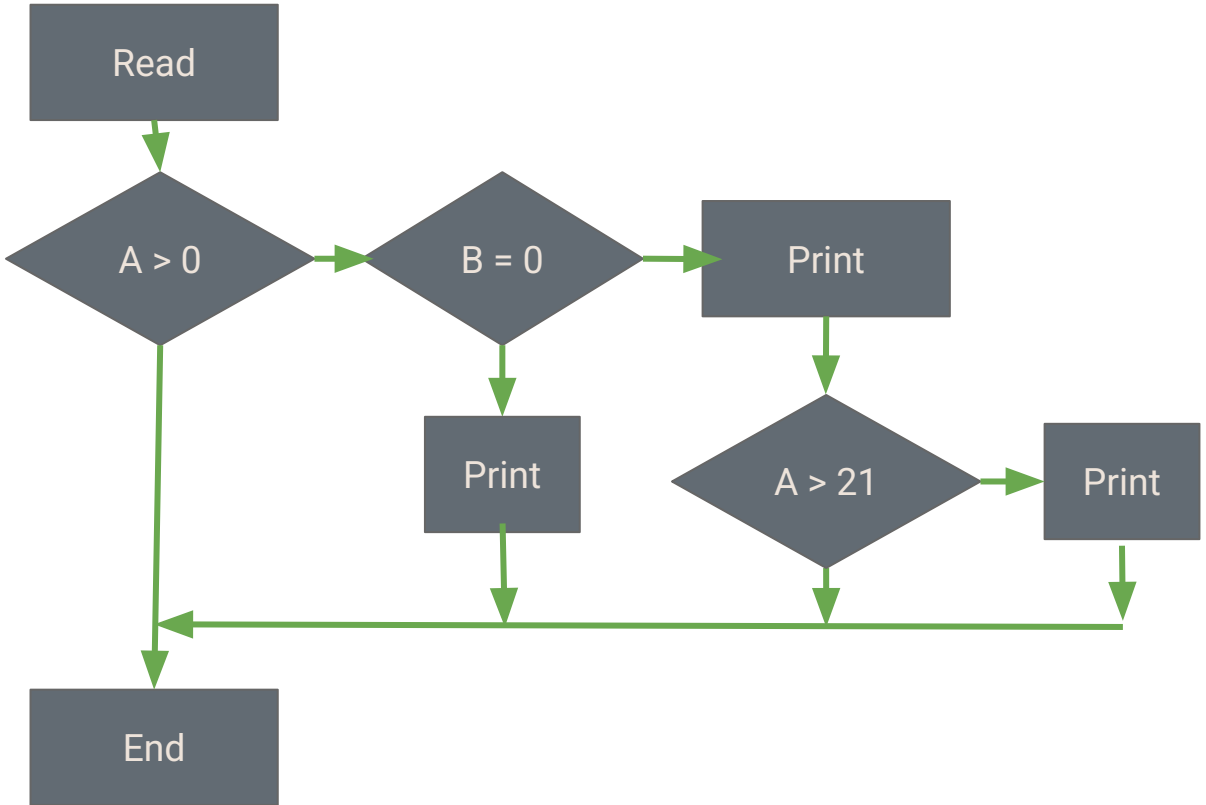


Приклад #3

```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
        IF A > 21 THEN
                Print A
        ENDIF
        ENDIF
ENDIF
```

Приклад #3

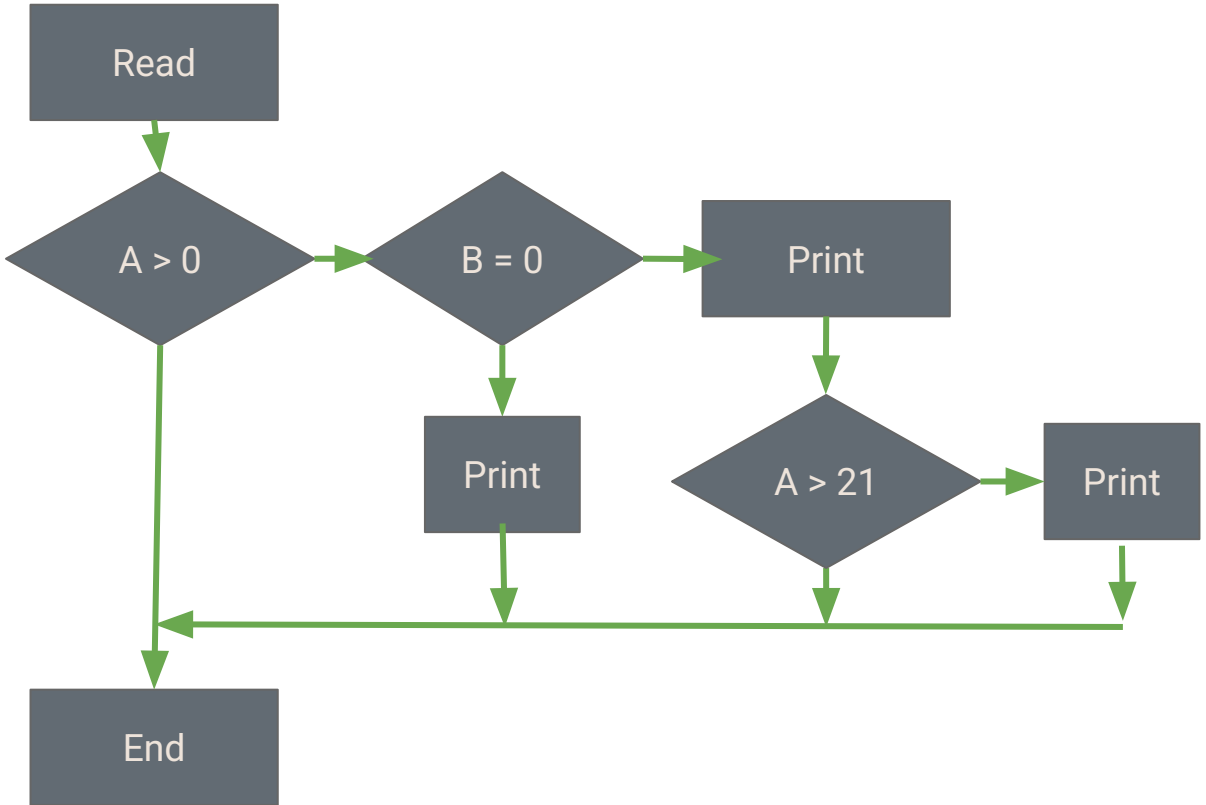
```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
        IF A > 21 THEN
                Print A
        ENDIF
        ENDIF
ENDIF
```



Приклад #3

```
Read A
Read B
IF A > 0 THEN
        IF B = 0 THEN
                Print “No values”
        ELSE
                Print B
        IF A > 21 THEN
                Print A
        ENDIF
        ENDIF
ENDIF
```

Decision coverage = 4



Приклад #4

```
Read A
Read B
IF A < 0 THEN
        Print “A negative”
ELSE
        Print “A positive”
ENDIF
IF B < 0 THEN
        Print “B negative”
ELSE
Print “B positive”
ENDIF
```

Приклад #4

Read A

Read B

IF A < 0 THEN

    Print “A negative”

ELSE

    Print “A positive”

ENDIF

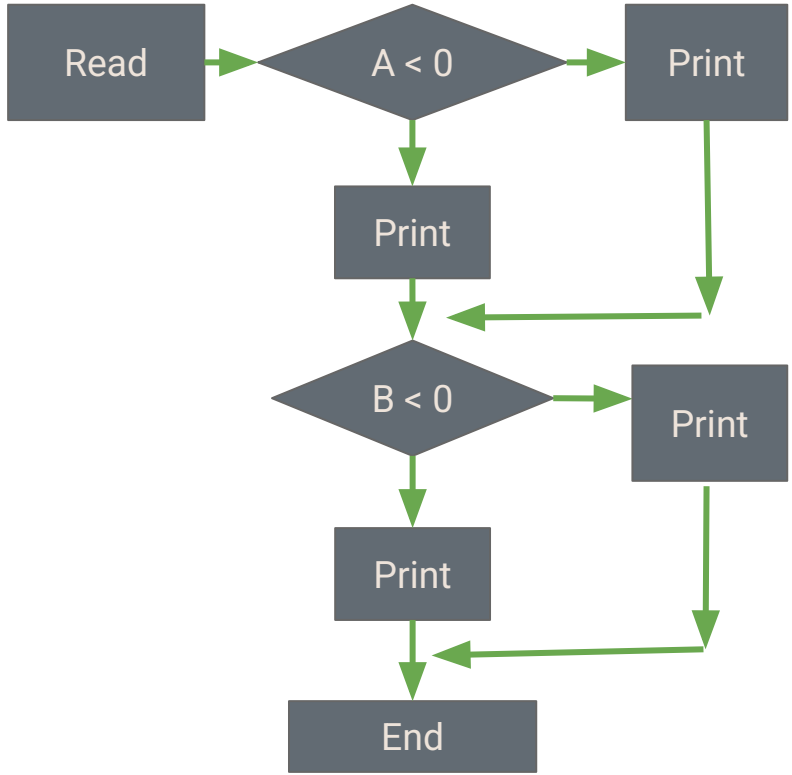
IF B < 0 THEN

    Print “B negative”

ELSE

Print “B positive”

ENDIF



Приклад #4

Read A

Read B

IF A < 0 THEN

    Print “A negative”

ELSE

    Print “A positive”

ENDIF

IF B < 0 THEN

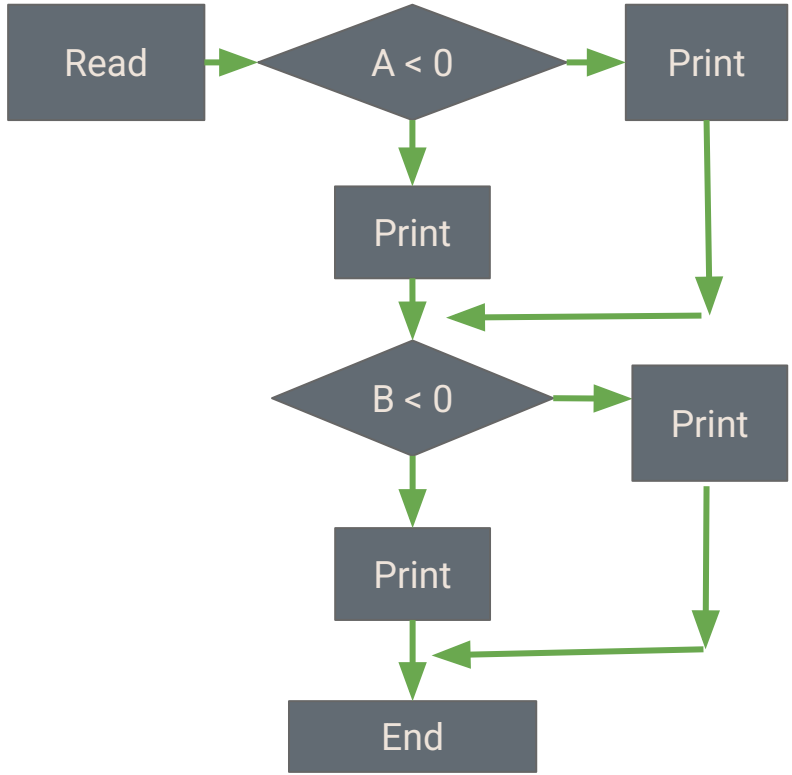
    Print “B negative”

ELSE

Print “B positive”

ENDIF

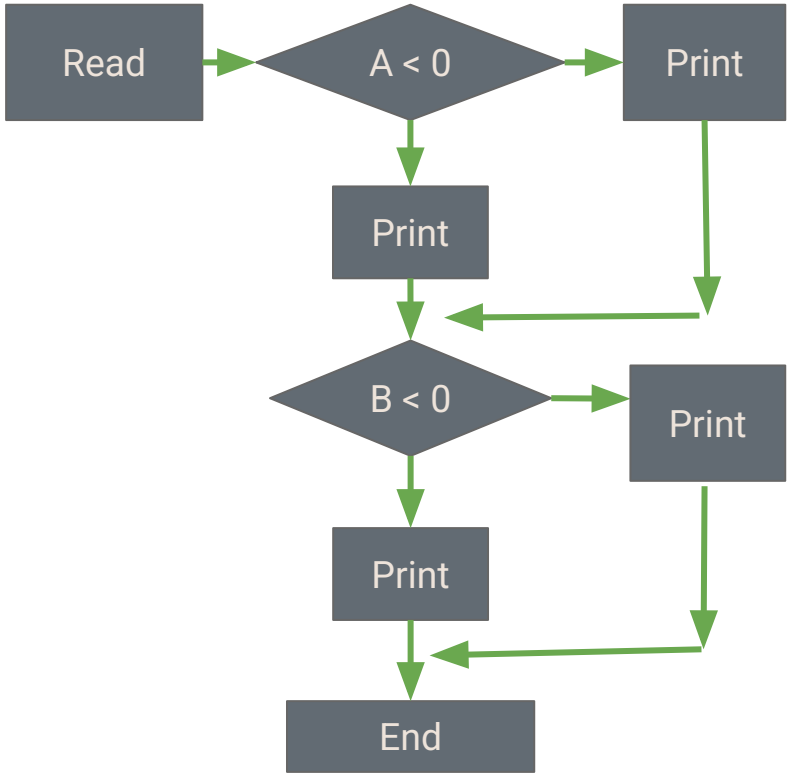
Decision coverage = 2



Приклад #4

```
Read A
Read B
IF A < 0 THEN
        Print “A negative”
ELSE
        Print “A positive”
ENDIF
IF B < 0 THEN
        Print “B negative”
ELSE
Print “B positive”
ENDIF
```

Decision coverage = 2



TIP: Завжди, незалежно від ELSE або IF Decision покриття = 2

Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке decision покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 62%
3. 75%
4. 100%

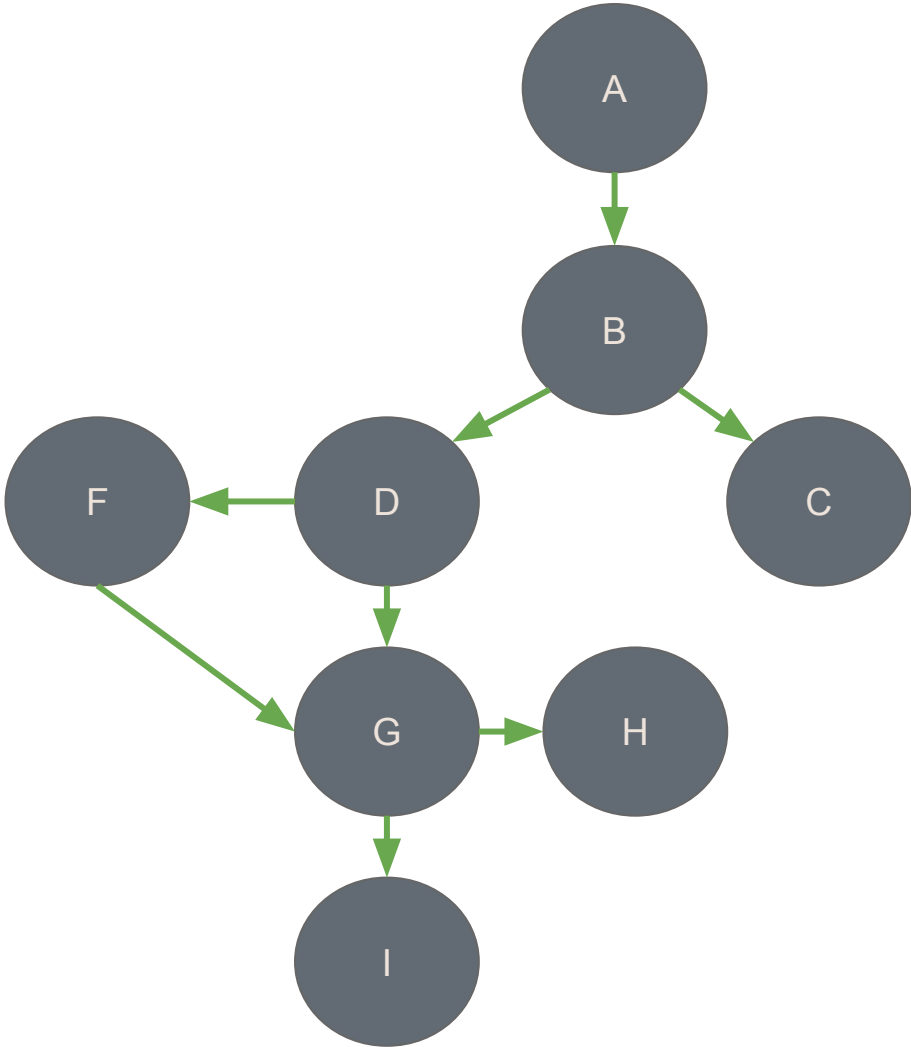
Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке decision покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 62%
3. 75%
4. 100%



Приклад #5

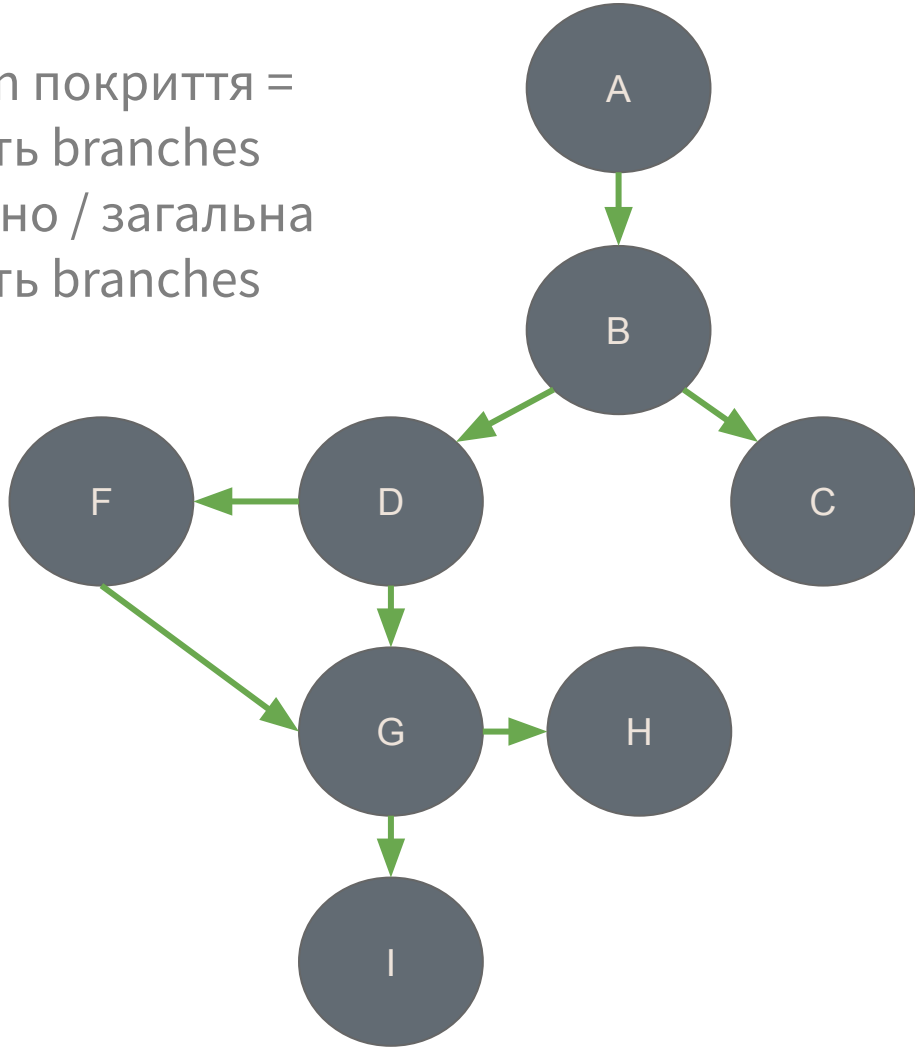
Для цього фрагмента коду було проведено такі шляхи/тести. Яке decision покриття досягнуто?

Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 62%
3. 75%
4. 100%

Decision покриття = кількість branches виконано / загальна кількість branches



Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке decision покриття досягнуто?

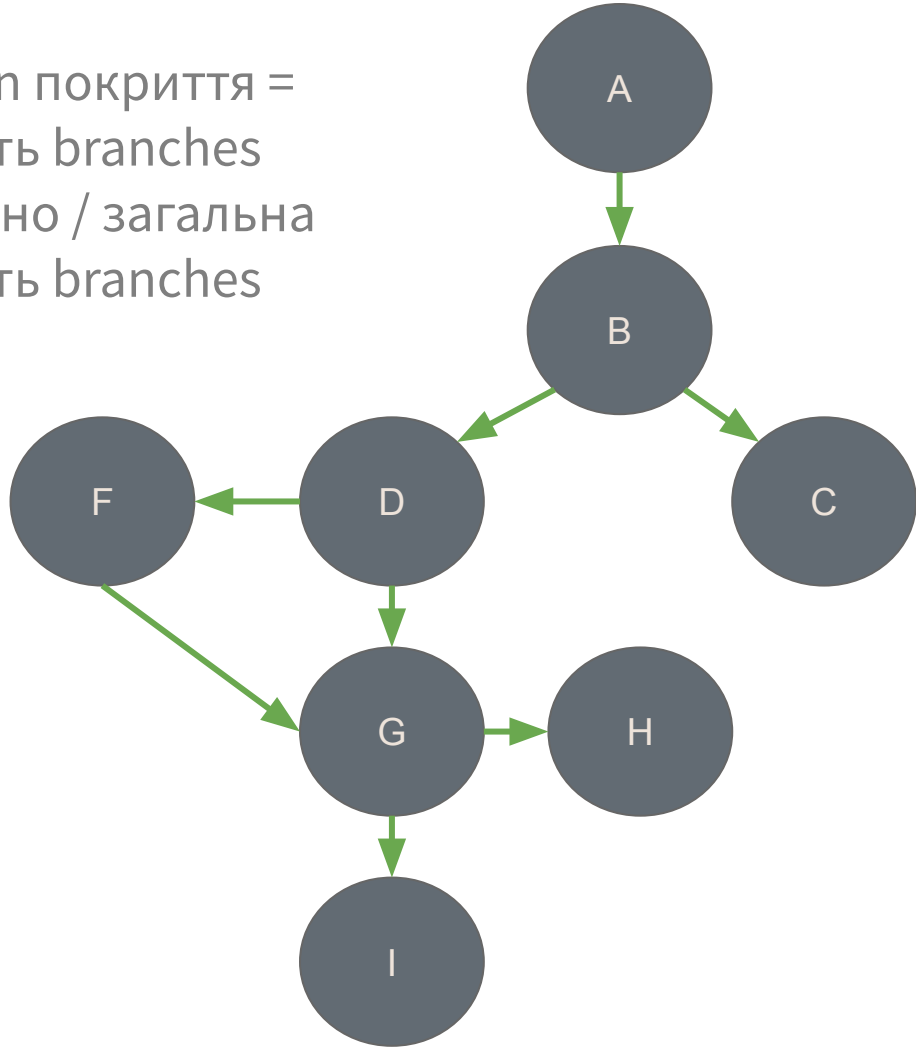
Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. 62%
3. 75%
4. 100%

5 із 8 branches перевірено

Decision покриття = кількість branches виконано / загальна кількість branches



Приклад #5

Для цього фрагмента коду було проведено такі шляхи/тести. Яке decision покриття досягнуто?

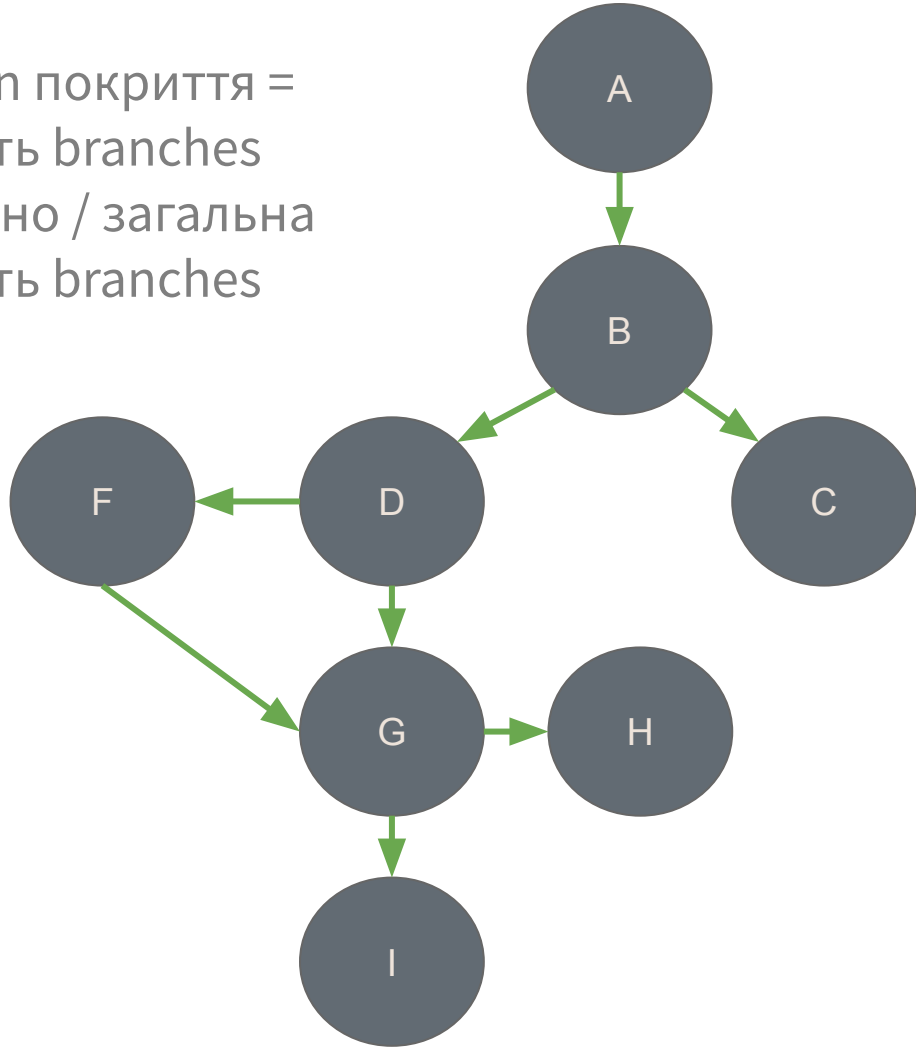
Тест 1 - A, B, C

Тест 2 - A, B, D, G, H

1. 50%
2. **62%**
3. 75%
4. 100%

5 із 8 branches перевірено

Decision покриття = кількість branches виконано / загальна кількість branches



Техніки, що ґрунтуються на досвіді

Техніки, засновані на досвіді

Техніки, засновані на досвіді, використовують уміння тестувальника, інтуїцію та досвід зі схожих додатків та технологій.

Зазвичай, ці техніки використовують після застосування формальних технік.

Є кілька сценаріїв, коли тільки техніки, що базуються на досвіді, можуть бути використані:

- Погані специфікації
- Мало часу на тестування
- Команда тестування погано розуміється на основах тестування

Вгадування помилок

- Ця техніка вимагає знання дефектів та досвід у додатку
- Потрібно описати можливі дефекти та створити тест кейси, які спробують знайти ці дефекти
- Список можливих дефектів може бути створений на основі досвіду, типових дефектів та даних про падіння
- Вгадування помилок ґрунтується на:
  - Як додаток працював у минулому
  - Які типи помилок можуть припуститися девелопери
  - Помилки, які були знайдені в подібних додатках

Дослідницьке тестування

- Дослідницьке тестування перевіряє частину програми, яку необхідно вивчити
- У дослідному тестуванні створюються неформальні тести, вони виконуються, створюється звіт і все це робиться у процесі тестування
- Тест кейси не вимагають підготовки та виконуються у рандомній або Ad-Hoc манері
- Виконання тестів обмежено проміжками часу від 30 до 120 хвилин, залежно від розміру програми
- Виконання тестів називають сесіями тестування у проміжках часу

Тестування, засноване на чек-лістах

- При тестуванні за чек-листами тестувальник проектує, реалізує та виконує тести, що покривають тестові умови, зазначені у чек-листі.
- Як складова аналізу тестувальники можуть створювати нові або розширювати чек-листи, або використовувати готові чек-листи, не змінюючи їх.
- Такі списки можуть бути побудовані на досвіді, на історичних даних про помилки, на інформації про пріоритети для користувачів та розуміння, як і чому відбуваються відмови у програмі.
- За відсутності детальних тестових сценаріїв, чек-листи допомагають визначити напрями тестування та збільшують узгодженість тестування.
- Оскільки чек-листи містять загальний опис, це знижує повторюваність результатів.