

## Комп'ютерний практикум №7

### Тестування методом чорного ящика

1. Теоретичні положення.....	1
2. Завдання.....	8
3. Вимоги до звіту.....	9

*Мета* – вивчити особливості застосування методів розбиття на класи еквівалентності та класи граничних значень при тестуванні методом чорного ящика.

#### 1.1 Теоретичні положення

При тестуванні методом чорного ящика програма представляється у вигляді «чорного ящику», для якого відомі первісні данні (X), результат роботи програми (Y) та функція програми. Мета тестування – виявлення обставин, при яких поведінка програми не відповідає специфікації.

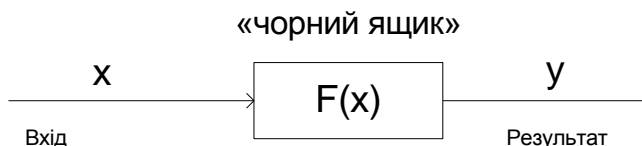


Рисунок 1

Для виявлення усіх помилок програми необхідно виконати безліч тестів – на всіх наборах даних. Проте для більшості класів програм такий підхід є неможливим. Тому необхідно виконати таке тестування програми, при якому буде охоплена лише невелика множина усіх наборів даних, причому обрана множина повинна мати найбільшу ймовірність виявлення помилок.

Тестування методом “чорного ящику” забезпечує пошук наступних класів помилок:

- некоректні чи відсутні функції;
- помилки інтерфейсу;
- помилки зовнішніх структур даних чи доступу до зовнішньої бази даних;
- помилки характеристик (наприклад, необхідність оперативної пам’яті);
- помилки ініціалізації та завершення.

До методів “чорного ящику” відносять:

- еквівалентне розбиття;
- аналіз граничних значень;
- аналіз причино-наслідкових зв’язків;
- припущення про помилку.

##### 1.1.1 Еквівалентне розбиття (класи еквівалентності)

Даний метод є найбільш розповсюдженим. Сутність методу полягає у наступному: вхідна множина даних розділяється на певні класи, для кожного класу визначається лише один тестовий варіант. *Клас еквівалентності* – це набір даних з однаковими властивостями. При опрацюванні різних елементів одного класу програма має вести себе однаково.

Таким чином, метод може бути сформульований наступним чином:

1. Початкові дані розбиваються на скінчену кількість класів еквівалентності. До одного класу еквівалентності належать такі тести, що, у випадку коли певний

тест класу виявляє помилку, то і будь-який інший тест цього ж класу має виявити ту ж помилку.

2. Кожен клас має містити якомога більшу кількість тестів, що дозволить мінімізувати загальну кількість тестів.

#### *Приклад*

Якщо у специфікації програми визначено, що допустимими вхідними даними є 3-розрядні числа у діапазоні 250 – 500, то можемо виділити такі три класи еквівалентності:

Клас 1 (допустимі вхідні дані) – числа від 250 до 500.

Клас 2 (недопустимі вхідні дані) – числа, що є меншими 250.

Клас 3 (недопустимі вхідні дані) – числа, що є більшими за 500.

Виявлення класів еквівалентності здійснюється евристичним методом, проте існують деякі правила:

1. Якщо вхідна умова описує деяку область значень, наприклад “ціле число, що приймає значення від 0 до 999”, то існує один правильний клас еквівалентності, та два – неправильних.
2. Якщо вхідна умова описує кількість значень, наприклад “кількість рядків у вхідному файлі може приймати значення від 1 до 6”, то існує один правильний клас еквівалентності, та два – неправильних.
3. Якщо вхідна умова описує множину вхідних значень, то визначається кількість правильних класів, яке дорівнює кількості елементів у множині вхідних значень.
4. Якщо вхідна умова описує ситуацію “повинно бути”, наприклад “перший символ має бути заголовним”, то існує один правильний клас та один неправильний.
5. Якщо має місце припущення, що елементи одного класу еквівалентності можуть оброблятися програмою по-різному, необхідно розбити клас на підкласи.

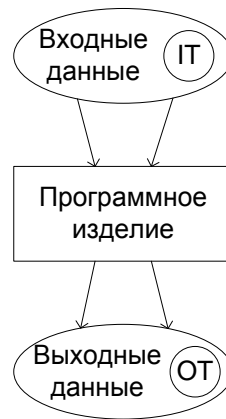
#### 1.1.2 Аналіз граничних значень

Головна ідея методу полягає в припущенні, що більша частина помилок виникає на границі області вводу. Тому при використанні методу граничних значень робота кожної з функцій тестується на усій області визначення. Розроблені тести повинні визначати перевірку правильності:

- виконання функцій програми;
- принципів обробки первісних даних;
- отримання результатів;
- забезпечення цілісності зовнішньої інформації;
- як отримуються результати.

Сукупність тестів дозволяє отримати комбінацію вхідних даних, які забезпечують повну перевірку усіх функціональних вимог до програми. Дана сукупність зазвичай складається з двох наборів:

- набір, що утворюється тими вхідними даними, що призводять до аномалії проведення програми (ІТ);
- набір, що утворюється такими кінцевими даними, що демонструють дефекти програми (ОТ).



**Рисунок 2**

Таким чином, граничні умови – це ситуації, що виникають на верхніх та нижніх границях вхідних класів еквівалентності. Відмінності даного методу від методу розбиття на класи еквівалентності полягають у наступному:

- вибір будь-якого елементу у класі еквівалентності у якості представника класу здійснюється таким чином, щоб одним тестом перевірити кожну границю цього класу;
- при створенні тестів розглядаються не лише вхідні дані (простір входів), але кінцеві (простір результатів).

Правила створення тестів:

1. Побудувати тести з невірними вхідними даними для ситуації несуттєвого виходу за границі області визначення.

*Наприклад*, якщо вхідні значення мають бути в інтервалі  $[-1.0 .. +1.0]$ , то перевіряємо  $-1.0, 1.0, -1.000001, 1.000001$ .

2. Обов'язково створювати тести для мінімальної та максимальної границі діапазону.
3. Якщо вхід та вихід програми являють собою впорядковану множину, зосередити увагу на першому та останньому елементах списку.

Слід зазначити, що при правильному застосуванні метод дозволяє виявити велику кількість помилок. Проте метод не може застосовуватись для перевірки комбінацій вхідних значень.

### 1.1.3 Аналіз причинно-наслідкових зв'язків


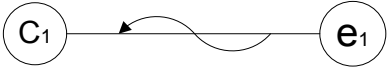
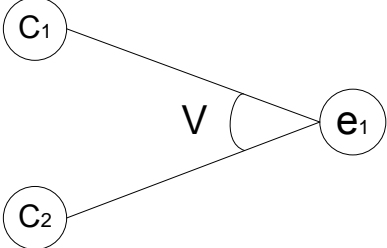
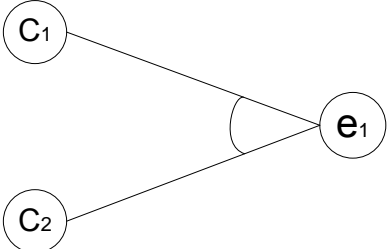
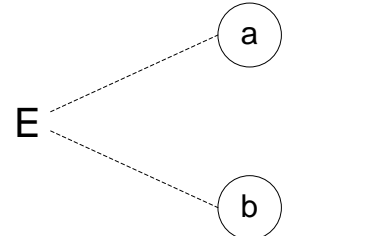
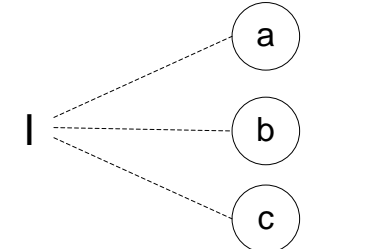
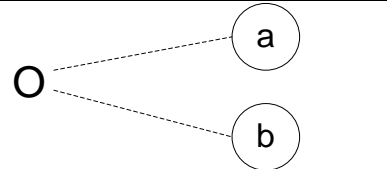


Для даного методу причина – зокрема вхідна умова чи клас еквівалентності. Наслідок – кінцева умова чи перетворення програми. Кожній причині та кожному наслідку привласнюється унікальний номер.

Етапи побудови тесту:

1. Розбиття специфікації програми на ділянки.
2. За допомогою аналізу семантичного змісту специфікації будується таблиця істинності, у якій послідовно переглядаються всі комбінації причин ( $c_i$ ) та визначаються наслідки ( $e_i$ ) для кожної комбінації.
3. Таблиця доповнюється примітками, які визначають обмеження та неможливі комбінації.

Недоліком даного методу є погане дослідження граничних умов.

При побудові таблиці істинності додатково може будуватись граф програми. Кожен вузол графу може знаходитись у стані 0 – стан відсутній, чи 1 – присутній.

Найменування функції	Графічне позначення	Пояснення
Тотожність		Якщо значення $c_1 = 1$ , то $e_1 = 1$ , якщо ж $c_1 = 0$ , то $e_1 = 0$
НІ		Якщо значення $c_1 = 1$ , то $e_1 = 0$ , в протилежному випадку $e_1 = 1$
АБО		Якщо $c_1$ або $c_2$ дорівнюють 1, то $e_1 = 1$ , в протилежному випадку $e_1 = 0$ .
І		Якщо $c_1$ і $c_2$ дорівнюють 1, то $e_1 = 1$ , в протилежному випадку $e_1 = 0$ .
Обмеження E		E має бути істинним, якщо хоча б одна з величин a чи b приймає значення 1 (одночасно a і b не можуть дорівнювати 1).
Обмеження I		Хоча б одна з величин a, b чи c завжди має дорівнювати 1.
Обмеження O		Лише одна з величин a чи b має дорівнювати 1.
Обмеження R		Якщо a приймає значення 1, то й b має приймати значення 1.
Обмеження M		Якщо наслідок a приймає значення 1, то й наслідок b має приймати значення 1.

#### 1.1.4 Припущення про помилку

Даний метод заснований на досвіді та інтуїції програміста. Ідея методу – створення переліку, що перераховує всі можливі помилки та ситуації, в яких можуть проявитися дані помилки. На основі переліку створюються тести.

## 1.2 Приклад застосування методів тестування

### 1.2.1 Еквівалентне розбиття (класи еквівалентності)

Постановка задачі – розробити тести для перевірки працездатності програми, що обчислює значення функції  $\cos(x)$  за допомогою розкладення у ряд Тейлора.

#### Етап 1. Створення специфікації програми

*Вхідні дані:*

$X$  – вхідний аргумент (у градусах): ціле число, діапазон значень від 0 до 30;

$e$  – точність: ціле число,  $e > 0$ .

*Результат:*

$S$  – значення обчислень: дійсне число,  $-1 \leq S \leq 1$ .

*Передумови:* вхідні дані входять до заданого діапазону.

*Постумови:* програма повертає знайдене значення  $\cos(x)$ .

#### Етап 2. Визначення класів еквівалентності шляхом побудови дерева розбиття

На наступному рисунку приведені дерево розбиття на класи еквівалентності:

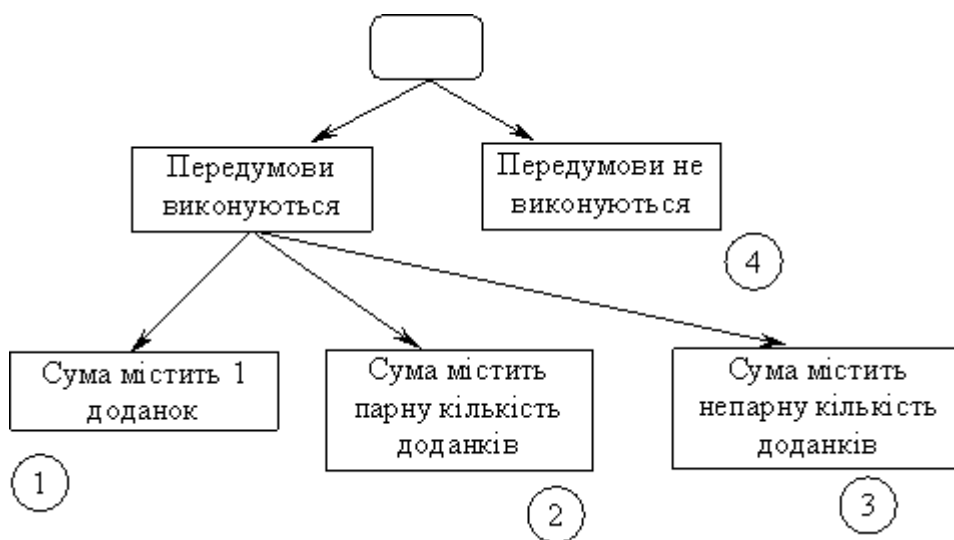


Рисунок 3

#### Етап 3. Складання тестових варіантів

У таблиці приведені тестові варіанти, складені відповідно до класів еквівалентності:

**Таблиця** Ошибка! Текст указанного стиля в документе отсутствует. **1 – Тестові варіанти (ТВ)**

ТВ	Вхідні дані	Очікуваний результат
1	$X = 30, e = 1$	1
2	$X = 30, e = 0.01$	0,8661
3	$X = 30, e = 0.00001$	0,86602
4	$X = 120, e = -1$	Повідомлення про помилку
5	$X = a, e = 1$	Повідомлення про помилку
6	...та інші варіанти	

### 1.2.2 Аналіз граничних значень

**Постановка задачі.** Нехай необхідно протестувати програму бінарного пошуку. Пошук виконується у масиві  $M$ , повертається індекс  $I$  елементу масива, значення якого відповідає ключеві пошуку  $K$ .

*Передумови:*

- 1) масив є впорядкованим;
- 2) масив має не менше одного елементу;
- 3) нижня межа масиву менше чи дорівнює верхній межі.

Післяумови:

- 1) якщо елемент знайдений, то пошук є вдалим (Result = True), I – номер елементу;
- 2) якщо елемент не знайдений, то пошук є невдалим (Result = False), I – не визначено.

Для формування класів еквівалентності необхідно провести розбиття області ІД – побудувати дерево розбиття (кожен лист дерева визначатиме окремий тестовий варіант):

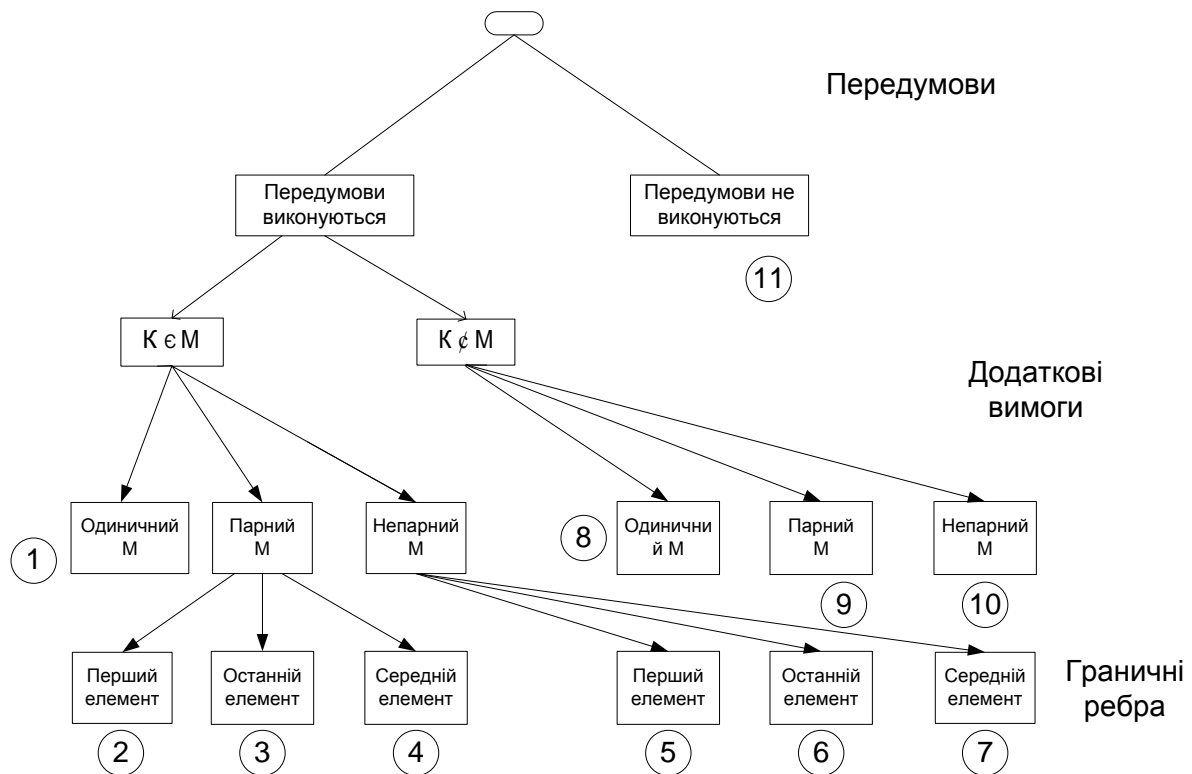


Рисунок 4 – Дерево розбиття області даних бінарного пошуку

Приклади тестових варіантів приведені в таблиці (таблиця 1.1).

**Таблиця** Ошибка! Текст указанного стиля в документе отсутствует..1 – Приклади тестових варіантів

№ТВ	ІД	Очікуваний результат
ТВ1	M = 10; K = 10	Result = True; I = 1
ТВ 2	M = (2, 7, 10, 17,28, 40); K = 2	Result = True; I = 1
ТВ 3	M = (2, 7, 10, 17,28, 40); K = 40	Result = True; I = 6
ТВ 4	M = (2, 7, 10, 17,28, 40); K = 17	Result = True; I = 4
ТВ 5	M = (2, 7, 10, 17,28, 40, 51); K = 2	Result = True; I = 1
ТВ 6	M = (2, 7, 10, 17,28, 40, 51); K = 51	Result = True; I = 7
ТВ 7	M = (2, 7, 10, 17,28, 40, 51); K = 28	Result = True; I = 5
ТВ 8	M = 10; K = 11	Result = False ; I = ?
ТВ 9	M = (2, 7, 10, 17,28, 40); K = 51	Result = False ; I = ?
ТВ 10	M = (2, 7, 10, 17,28, 40, 51); K = -3	Result = False ; I = ?
ТВ 11	M = (28, 2, 10, 17, 40, 7); K = 2 <sup>8</sup>	Аварійне повідомлення: масив не впорядкований

### 1.2.3 Аналіз причинно-наслідкових зв'язків

**Постановка задачі.** Програма виконує розрахунок сплати послуг за електроенергію за середнім чи змінним тарифом.

При розрахунку за середнім тарифом:

- при місячному споживанні енергії менш, ніж 100 кВт/год. розраховується фіксована сума;
- при споживанні енергії  $\geq 100$  кВт/год., застосовується процедура А планування розрахунку.

При розрахунку за змінним тарифом:

- при місячному споживанні енергії менш, ніж 100 кВт/год. застосовується процедура А планування розрахунку;
- при споживанні енергії  $\geq 100$  кВт/год., застосовується процедура В планування розрахунку.

#### Крок 1. Визначення причин та наслідків

Причинами для даної задачі є:

- 1) розрахунок за середнім тарифом;
- 2) розрахунок за змінним тарифом;
- 3) місячне споживання електроенергії менше, ніж 100 кВт/год.;
- 4) місячне споживання електроенергії  $\geq 100$  кВт/год.

Наслідки:

- 101 – фіксована місячна вартість (мінімальна);
- 102 – процедура А планування розрахунку;
- 103 – процедура В планування розрахунку.

#### Крок 2. Побудова графу причинно-наслідкових зв'язків

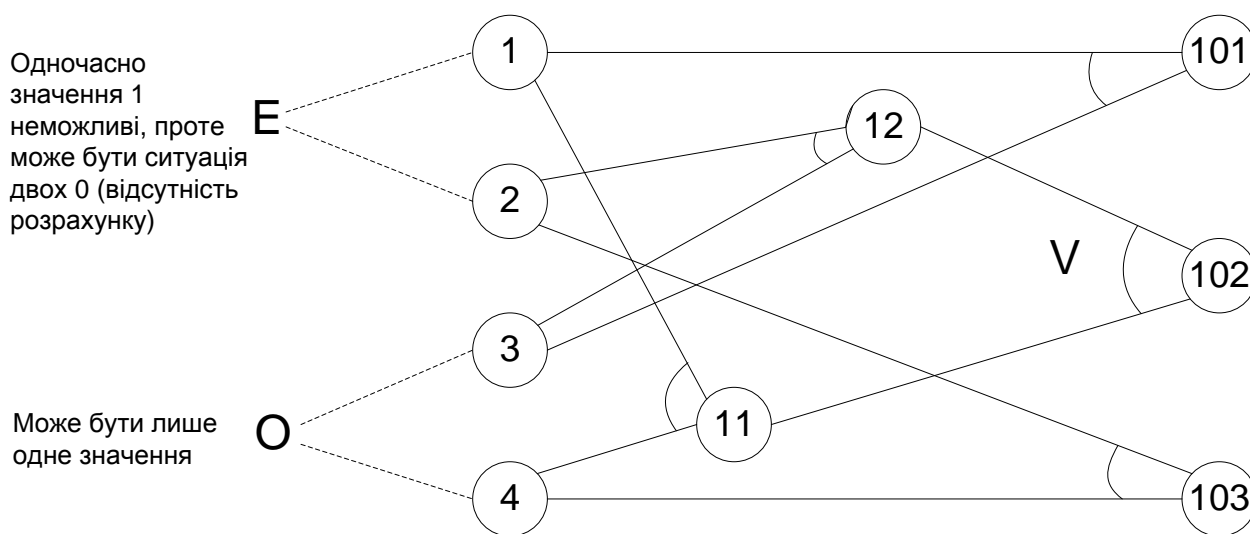


Рисунок 5

#### Крок 3. Побудова матриці

Порядок заповнення матриці

1. Обираємо деякий наслідок, який має бути у стані 1.
2. Знаходяться всі комбінації причин (із урахуванням обмежень), які встановлюють цей наслідок у 1 (зворотній прогін графу).
3. Для кожної комбінації причин, що приводять наслідок у стан 1, будується один стовпець.
4. Для кожної комбінації причин визначаються стани всіх інших наслідків, що розміщуються в той самий стовпець таблиці.
5. Кроки 1–4 повторюються для всіх наслідків графу.

			Номера стовпців			
			1	2	3	4
Умови (причини)	причини	1	1	0	1	0
		2	0	1	0	1
		3	1	1	0	0
		4	0	0	1	1
	вторинні причини	11	0	0	1	1
		12	0	0	1	0
Дія	наслідок	101	1			
		102		1	1	
		103				1

#### Крок 4. Перетворення стовпців до тестових варіантів (ТВ)

Згідно із кроком 3 маємо 4 тестових варіанти

ТВ	Початкові дані	Очікуваний результат
1	Розрахунок за середнім тарифом, споживання = 60 кВт/год.	Фіксована мінімальна вартість
2	Розрахунок за змінним тарифом, споживання = 90 кВт/год.	Процедура А планування розрахунку
3	Розрахунок за середнім тарифом, споживання = 100 кВт/год.	Процедура А планування розрахунку
4	Розрахунок за змінним тарифом, споживання = 100 кВт/год.	Процедура В планування розрахунку

#### 1.3 Завдання

Для заданої згідно із варіантом постановки задачі виконати наступну послідовність дій:

1. Скласти специфікацію на програмний код.
2. Розробити програмний код.
3. Визначити класи еквівалентності шляхом побудови дерева розбиття.
4. Скласти тестові варіанти.
5. Виконати тестування.
6. Оформити результати тестування.

#### Варіанти завдань

Визначити значення функції у, при цьому розкласти функцію **f(x)** в ряд Тейлора (Маклорена). Значення  $x$ ,  $n$  і  $m$  визначаються користувачем.

В-т	y	f(x)	В-т	y	f(x)	В-т	y	f(x)
1	$\frac{n!-m!}{n-m} + \sin(x)$	$\sin(x)$	11	$\frac{\sqrt{x^2-4}}{x}$	$\sqrt{x}$	21	$\frac{x+1}{sh(x)+x}$	$sh(x)$
2	$\frac{\ln(x)}{\ln(x-2)}$	$\ln(x)$	12	$\frac{e^x-x}{x}$	$e^x$	22	$\frac{tgx+tg(2x)}{x+2}$	$tg(x)$
3	$\frac{1}{\sqrt{x^2-1}}$	$\sqrt{x}$	13	$\cos(x+2) + \frac{1}{x+1}$	$\cos(x)$	23	$\sqrt{x^2-1}$	$\sqrt{x}$



В- т	y	f(x)	В- т	y	f(x)	В- т	y	f(x)
4	$\frac{\cos(x/2)}{\cos(x+\frac{\pi}{2})}$	cos(x)	14	$\frac{\sin(x+1)}{\sin^2(x)}$	sin(x)	24	$\cos(\frac{1}{x+1}+\frac{\pi}{4})$	cos(x)
5	$\frac{tg(x^2)-2 \cdot x}{tg(x)}$	tg(x)	15	$arctg^2(x+1)-\frac{n!}{n-m}$	arctg(x)	25	$e^x+\frac{1}{x^2-1}$	e <sup>x</sup>
6	$\frac{arctg(x)}{arctg(x-5)}$	arctg(x)	16	$arctg(\frac{1}{x-1})+x$	arctg(x)	26	$\sin(x)+\frac{1}{x!-1}$	sin(x)
7	$\frac{n-m}{n!-m!}+e^x$	e <sup>x</sup>	17	$\frac{sh(x)}{sh(x^2)}$	sh(x)	27	$arctg(x)+arctg(\frac{1}{x})$	arctg(x)
8	$\frac{x}{\sqrt{x^2-5x+6}}$	$\sqrt{x}$	18	$\frac{x-3}{\ln(x)}$	ln(x)	28	$\frac{sh(x+2)}{sh(x+1)}$	sh(x)
9	$\frac{x+3}{\sin(x)}$	sin(x)	19	$\frac{\cos(x-2)}{x \cdot \cos(x)}$	cos(x)	29	$\frac{x}{\sqrt{x^2+3x+2}}$	$\sqrt{x}$
10	$\frac{1}{tg(x)+tg(x/2)}$	tg(x)	20	$\frac{sh^2(x)}{sh(x+2)}$	sh(x)	30	$tg(x+\frac{\pi}{4})$	tg(x)

#### 1.4 Вимоги до звіту

Звіт має містити такі розділи:

1. ПІБ виконавця та номер варіанту.
2. Опис постановки задачі.
3. Специфікація на програму (передумови та постумови).
4. Дерево розбиття.
5. Тестові варіанти.
6. Результати тестування.