Лекція 5. Технічне завдання

ДСТУ 19.201-78 «Технічне завдання. Вимоги до змісту та оформленню»

Стандарт встановлює порядок побудови і оформлення технічного завдання на розробку програми або програмного виробу для обчислювальних машин, комплексів та систем незалежно від їх призначення та області застосування.

ТЗ має містити наступні розділи:

- Вступ
- Підстави для розробки
- Призначення розробки
- Вимоги до програми
- Вимоги до програмної документації
- Техніко-економічні показники
- Стадії та етапи розробки
- Порядок контролю і прийому
- **Додатки (при потребі)**

ЗМІСТ РОЗДІЛІВ

1. В розділі «***Вступ***» вказують найменування, коротку характеристику області застосування програми і об'єкту, в якому будуть застосовувати програму
2. В розділі «***Підстави для розробки***» мають бути вказані:

- Документи. На підставі яких ведеться розробка
- Організація, яка затвердила цей документ і дата затвердження
- Найменування і умовне позначення теми розробки

3. «***Призначення розробки***» - функціональне і експлуатаційне призначення програми

4. Розділ «***Вимоги до програми***» має містити наступні підрозділи:

- Вимоги до функціональних характеристик – вимоги до складу виконуваних функцій, організації вхідних та вихідних даних, часовим характеристикам тощо…
- Вимоги до надійності – вимоги до забезпечення надійного функціонування (контроль вхідної та вихідної інформації, час відновлення після збоїв)
- Умови експлуатації – температура повітря, вологість та інші для вибраного типу носіїв даних, при яких мають забезпечуватись задані характеристики, а також вид обслуговування, кількість та кваліфікацію персоналу;

4. Розділ «***Вимоги до програми***» має містити наступні підрозділи:

- Вимоги до складу і параметрам технічних засобів – вказують необхідний склад технічних засобів із вказанням їх основних технічних характеристик
- Вимоги до інформаційної та програмної сумісності – вимоги до інформаційних структур на вході та виході, методам розв'язку, кодам програми, мовам програмування і програмним засобам, які використовує програма. При необхідності має забезпечуватись захист інформації та програм
- Вимоги до маркування та пакування
- Вимоги до транспортування та зберігання
- Спеціальні вимоги

В розділі «***Вимоги до програмної документації***» має бути вказаний попередній склад програмної документації і спеціальні вимоги до неї

В розділі «***Техніко-економічні показники***» мають бути вказані:

- Орієнтовна економічна ефективність
- Приблизна річна потреба в програмі
- Економічні переваги розробки у зрівнянні з аналогами

В розділі «***Стадії та етапи розробки***» встановлюють необхідні стадії розробки, етапи і зміст робіт (перелік програмних документів, які мають бути розроблені, узгоджені і затверджені), строки розробки і виконавців

В ***додатках до ТЗ*** за необхідністю приводять:

- Перелік науково-дослідницьких та інших робіт, які пояснюють необхідність розробки
- Схеми алгоритмів, таблиці, розрахунки та інші документи, які можуть бути використані при розробці
- Інші джерела розробки

Тестування “білої скрині”

Програміст пише програми і сам їх тестує.

Ця технологія називається тестуванням «білої скрині» (white box) або «скляної скрині» (glass box).

Переваги тестування «білої скрині»:

- **1. Направленість тестування**. Програміст може тестувати програму по частинах, розробляти спеціальні тестові підпрограми, які визивають модель, що тестується, і передають йому необхідні дані. Окремий модуль завжди легше протестувати саме як «білу скриню»
- **2. Повне охоплення коду**. Програміст завжди може визначити, які саме фрагменти коду працюють в кожному тесті. Він бачить які гілки залишились не протестованими і може підібрати умови при яких вони будуть виконані
- **3. Керування потоком**. Програміст завжди знає, яка функція має виконуватись в програмі наступною і яким має бути її поточний стан. Тут програміст може користуватись таким зручним засобом як відладчик

- **4. Відстежування цілісності даних**. Програмісту відомо, яка частина програми має змінювати кожен елемент даних. Відстежуючи стан даних, він може виявити такі помилки як зміна даних не тими модулями, їх неправильна інтерпретація або невдала організація. Програміст також може самостійно автоматизувати тестування.
- **5. Внутрішні граничні точки**. В коді видно ті граничні точки програми, які скриті від погляду тестерів «чорної скрині». Наприклад, для виконання певної дії можна використати декілька алгоритмів і невідомо який з них вибрав програміст. Переповнення буфера – програміст одразу може сказати при якій кількості даних виникне ця помилка.
- **6. Тестування, яке визначається вибраним алгоритмом**. Для тестування обробки даних, яка використовує складні обчислювальні алгоритми, можуть знадобитись спеціальні технології.

Тестування «білої скрині» розглядають як частину програмування. Програміст виконує цю роботу постійно.

Структурне тестування є одним з видів тестування «білої скрині». Головна ідея – вибір правильного програмного шляху.

Структурне тестування має потужну математичну теоретичну основу, але більшість тестерів використовують функціональне тестування.

Тестування програмних шляхів: *критерії охоплення*

Програмний шлях – послідовність команд, які може виконувати програма у процесі функціонування.

На практиці протестувати усі програмні шляхи дуже важко. Тому серед програмних шляхів виділяють ті, що необхідно протестувати обов'язково.

Для відбору використовуються *критерії охоплення* (coverage criteria). Їх також називають *логічними критеріями охоплення* або *критеріями повноти*.

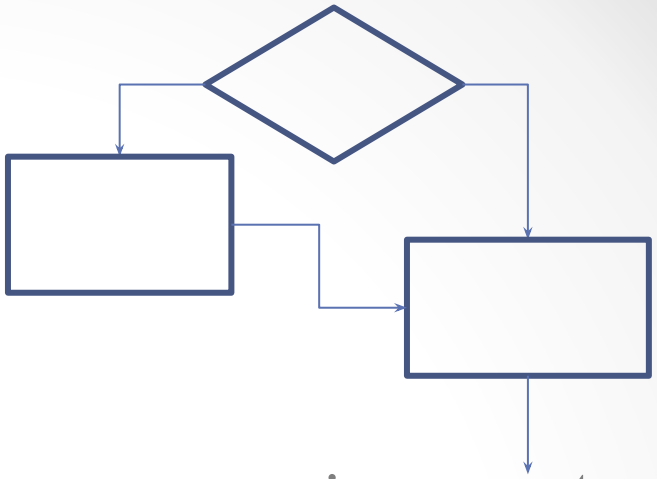
Найчастіше використовуються критерії:

- Охоплення рядків – кожний оператор має бути виконаний принаймні один раз.
- Охоплення розгалужень – для кожного розгалуження мають перевірятися усі його варіанти.
- Охоплення шляхів – мають бути перевірені усі програмні шляхи.
- Охоплення умов – мають бути перевірені усі складові частини кожної складної умови.

Коли тестування узгоджується з цими критеріями кажуть *про тестування шляхів*.

Фрагмент коду

```
if (a < b && c == 5)
    do something;
operator;
```



Для перевірки коду необхідно проаналізувати 4 варіанти:

- a < b and c = 5 : (do something виконується)
- a < b and c ≠ 5 : (do something не виконується)
- a ≥ b and c = 5 : (do something не виконується)
- a ≥ b and c ≠ 5 : (do something не виконується)

Для виконання критерію охоплення рядків необхідно перевірити тільки 1-й варіант

Критерій охоплення розгалуджень

Програміст перевіряє варіант 1 і ще один з трьох інших варіантів.

1. a < b and c = 5 : (do something виконується)
2. a < b and c ≠ 5 : (do something не виконується)
3. a ≥ b and c = 5 : (do something не виконується)
4. a ≥ b and c ≠ 5 : (do something не виконується)

Критерій охоплення умов

Необхідно перевірити усі 4 варіанти

Критерії охоплення даних

Критерії охоплення корисні але їх недостатньо.

Приклад

- **A = B/C**

Якщо C ≠ 0, той цей код успішно виконується і критерії охоплення виконуються. Але при C = 0 програма припинить виконання. Різниця в цих двох варіантах не в шляху, а в ***даних***.

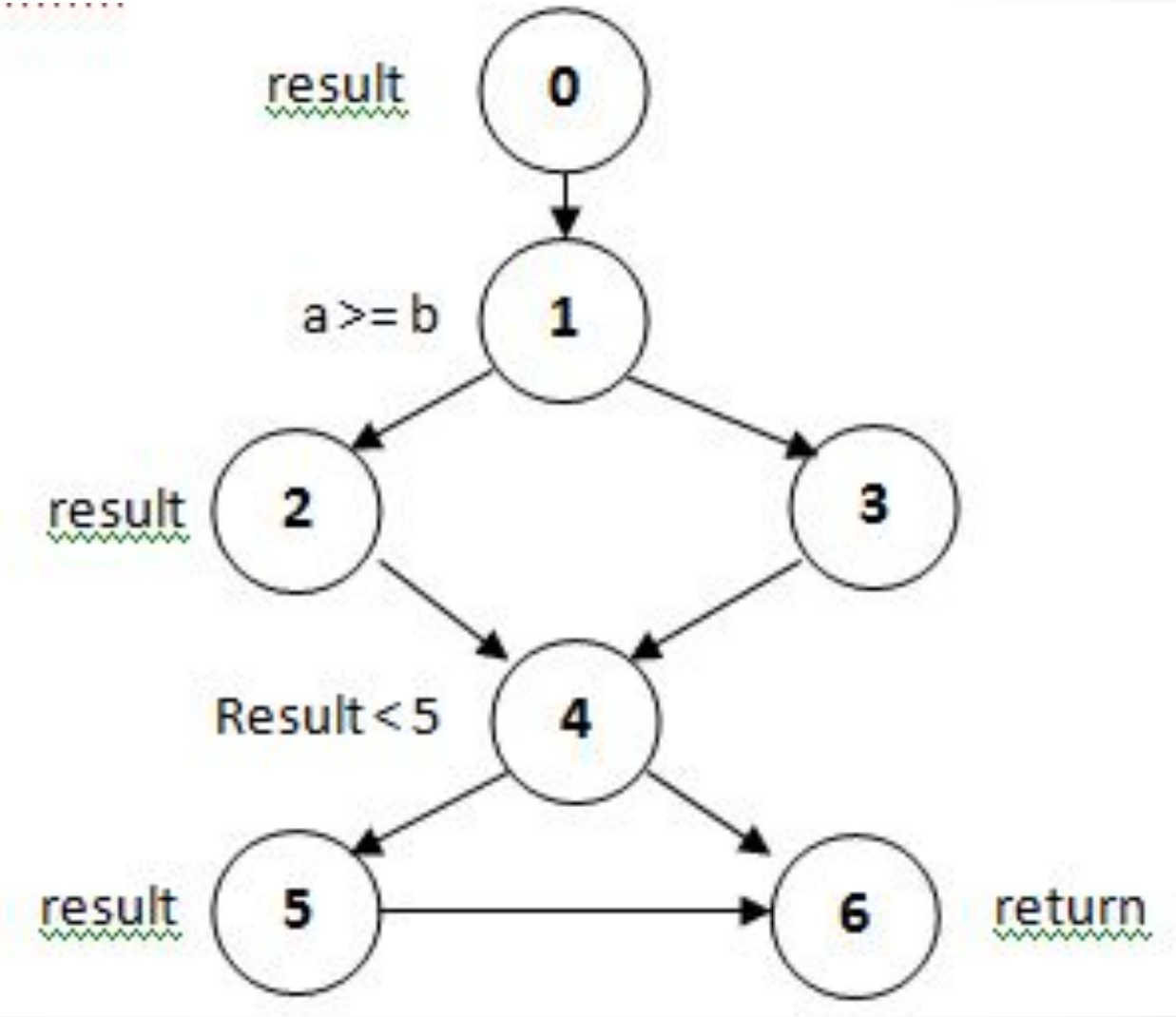
Програмний шлях називається *чутливим до помилок*, якщо при його проходженні можуть виникнути помилки.

Якщо ж помилки обов'язково виникають при проходженні даного шляху, то такий шлях називається *таким, що знаходить помилки*

Приклад тестування потоків керування

```
int f(int a, int b)
{
    int result = 0;
    if(a >= b)
     result = a-b;
    else
     result = b-a;
    if(result < 5)
     result = 0;
    return result;
}
```

Керуючий граф програми



Тестовий набір, сформований згідно з **критерієм покриття команд (рядків)**, має вигляд

$(a, b, f_{ет}) = \{(9,5,0), (5,9,0)\}$.

В протестованій по даному набору програмі виконуються усі команди і виконуються наступні шляхи:

$$M = \begin{cases} 0 \to 1 \to 2 \to 4 \to 5 \to 6 \\ 0 \to 1 \to 3 \to 4 \to 5 \to 6. \end{cases}$$

Тестовий набір, сформований згідно з **критерієм покриття розгалужень** має вигляд

$(a, b, f_{ет}) = \{(9, 5, 0), (0, 255, 255)\}$.

В тестованій по даному набору програмі виконуються усі розгалуження і виконується наступна множина шляхів:

$$M = \begin{cases} 0 \to 1 \to 2 \to 4 \to 5 \to 6 \\ 0 \to 1 \to 3 \to 4 \to 6. \end{cases}$$

Тестовий набір, сформований згідно з критерієм покриття маршрутів, має вигляд

$(a, b, f_{ет}) = \{(9, 5, 0), (0, 255, 255), (255, 0, 255), (5, 9, 0)\}$.

$$M = \begin{cases} 0 \to 1 \to 2 \to 4 \to 5 \to 6 \\ 0 \to 1 \to 3 \to 4 \to 6 \\ 0 \to 1 \to 2 \to 4 \to 6 \\ 0 \to 1 \to 3 \to 4 \to 5 \to 6. \end{cases}$$