МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна

«ЯКІСТЬ ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ ТА ТЕСТУВАННЯ**»**

Лабораторна робота № 10

«Низхідне тестування»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Завдання:

Отриманий кінцевий масив відсортувати за вказаним принципом – парні.
Виконання роботи передбачає наступну послідовність дій:

1. Створення (визначення) головного керуючого модуля;
2. Проектування заглушок кількох типів;
3. Сторення тестових варіантів;
4. Виконання тестування;
5. Оформлення результатів.

Звіт має містити:

1. Схему інтеграції;
2. Тексти заглушок;
3. Тестові варіанти;
4. Результати тестування у вигляді скріншотів.

Варіант 4
Даний цілочисельний масив A розміру N. Переписати в новий цілочисельний масив B всі парні числа з початкового масиву (у тому ж порядку) і вивести розмір отриманого масиву B і його вміст.

Виконання:

Приступаючи до лабораторної роботи з низхідного тестування, я почав зі створення головного модуля управління та розробки заглушок для необхідних функцій. Ось як я підійшов до кожного завдання:

1. **Створення головного модуля управління:**
   Я створив функцію з назвою main, яка слугувала головним модулем управління. Ця функція викликала інші функції, відповідальні за введення початкових даних, обробку даних і виведення результату.

```
def main():
    # Input the initial data
    A, N = input_data()

    # Process the data: filter even numbers
    B, size_B = filter_even_numbers(A, N)

    # Output the result
    output_result(B, size_B)

def input_data():
    # Stub for input_data function
    N = 5
```

```
    A = [1, 2, 3, 4, 5]
    return A, N

def filter_even_numbers(A, N):
    # Stub for filter_even_numbers function
    B = [2, 4]
    size_B = len(B)
    return B, size_B

def output_result(B, size_B):
    # Stub for output_result function
    print(f"Size of the resulting array B: {size_B}")
    print("Contents of array B:", B)

if __name__ == "__main__":
    main()
```

2. **Розробка заглушок для наступних функцій:**
    a. input_data: Я розробив заглушку для цієї функції, яка повинна зчитувати від користувача розмір N та цілочисельний масив A.
    b. filter_even_numbers: Я розробив заглушку для цієї функції, яка повинна отримати вхідний масив A і його розмір N, створити новий масив B, що містить тільки парні числа з A, і повернути розмір і вміст масиву B.
    c. sort_even_numbers: Я розробив заглушку для цієї функції, яка повинна сортувати парні числа в масиві B.
    d. output_result: Я розробив заглушку для цієї функції, яка повинна виводити розмір та вміст масиву B.

```
def input_data():
    # Stub for input_data function: Replace this with the actual implementation
    N = 5
    A = [1, 2, 3, 4, 5]
    return A, N

def filter_even_numbers(A, N):
    # Stub for filter_even_numbers function: Replace this with the actual implementation
    B = [2, 4]
    size_B = len(B)
    return B, size_B

def sort_even_numbers(B):
```

```
    # Stub for sort_even_numbers function: Replace this with the actual
implementation
    sorted_B = [2, 4]
    return sorted_B

def output_result(B, size_B):
    # Stub for output_result function: Replace this with the actual implementation
    print(f"Size of the resulting array B: {size_B}")
    print("Contents of array B:", B)
```

3. **Постановка тестових варіантів:**
   Я підготував тестові приклади, які охоплюють різні сценарії для вхідного масиву A. Приклади додаються:
   a. Масив A, що містить лише парні числа.
   b. Масив A, що містить лише непарні числа.
   c. Масив A, що містить суміш парних і непарних чисел.
   d. Масив A, що містить як додатні, так і від'ємні числа.
   e. Масив A, що має розмір 1.
   f.

   Нижче наведено підготовлені тестові приклади, які охоплюють різні сценарії для вхідного масиву A:
   g. Масив A містить лише парні числа:
      i. Вхідні дані: N = 5, A = [2, 4, 6, 8, 10].
      ii. Очікуваний результат: Розмір B = 5, B = [2, 4, 6, 8, 10].
   h. Масив A містить лише непарні числа:
      i. Вхідні дані: N = 5, A = [1, 3, 5, 7, 9].
      ii. Очікуваний вихід: Розмір B = 0, B = [].
   i. Масив A містить суміш парних та непарних чисел:
      i. Вхідні дані: N = 6, A = [1, 2, 3, 4, 5, 6].
      ii. Очікуваний вихід: Розмір B = 3, B = [2, 4, 6].
   j. Масив A містить як додатні, так і від'ємні числа:
      i. Вхідні дані: N = 7, A = [-3, 2, -1, 4, 5, -6, 8].
      ii. Очікуваний вихід: Розмір масиву B = 4, B = [2, 4, -6, 8].
   k. Масив A має розмір 1:
      i. Вхідні дані: N = 1, A = [3].
      ii. Очікуваний вихід: Розмір B = 0, B = [].

   Ці тестові приклади можна використовувати для оцінки коректності вашої програми, коли ви реалізуєте реальні функції, замінивши заглушки.

4. **Виконання тестування:**
   Я запустив основний модуль з підготовленими тестовими прикладами, використовуючи розроблені заглушки. Потім я порівняв результат роботи

програми з очікуваним результатом для кожного тесту.
Схема інтеграції: Схема, що показує основний модуль управління та його зв'язки з іншими функціями.
Ось приклад на Python, як можна модифікувати головний модуль і заглушки, щоб приймати вхідні дані з підготовлених тестових кейсів за допомогою циклу:

```
def main(test_case):
    N, A = input_data(test_case)
    size, B = filter_even_numbers(N, A)
    B = sort_even_numbers(B)
    output_result(size, B)

# Test cases
test_cases = [
    (5, [2, 4, 6, 8, 10]),
    (5, [1, 3, 5, 7, 9]),
    (6, [1, 2, 3, 4, 5, 6]),
    (7, [-3, 2, -1, 4, 5, -6, 8]),
    (1, [3])
]

# Execute testing using a loop
for test_case in test_cases:
    print(f"Testing with input: {test_case}")
    main(test_case)
    print()
```

У цьому прикладі функція main викликається всередині циклу, який повторюється для кожного тесту. Заглушка input_data модифікується, щоб прийняти тестовий кейс як вхідні дані, а жорстко закодовані значення в заглушках замінюються на вхідні дані тестового кейсу.

a. Схема інтеграції: Створіть схему, що показує основний модуль управління та його зв'язки з іншими функціями, такими як input_data, filter_even_numbers, sort_even_numbers та output_result.
b. Тестові кейси: Я перерахував тестові кейси, використані для тестування, включаючи вхідні дані та очікуваний результат.
    i. Вхідні дані: (5, [2, 4, 6, 8, 10])
    ii. Очікуваний результат: Розмір B = 5, B = [2, 4, 6, 8, 10].
    iii. Тест 2:

    iv. Вхідні дані: (5, [1, 3, 5, 7, 9])
    v. Очікуваний результат: Розмір B = 0, B = [].
    vi. Тестовий приклад 3:

vii. Вхідні дані: (6, [1, 2, 3, 4, 5, 6])
viii. Очікуваний результат: Розмір B = 3, B = [2, 4, 6].
ix. Тестовий приклад 4:

x. Вхідні дані: (7, [-3, 2, -1, 4, 5, -6, 8])
xi. Очікуваний результат: Розмір B = 4, B = [-6, 2, 4, 8].
xii. Тест 5:

xiii. Вхідні дані: (1, [3])
xiv. Очікуваний результат: Розмір B = 0, B = [].

c. Результати тестування у вигляді скріншотів: Я додав скріншоти результатів тестування, що показують вивід програми для кожного тестового випадку.

```python
# Test cases
test_cases = [
    (5, [2, 4, 6, 8, 10]),
    (5, [1, 3, 5, 7, 9]),
    (6, [1, 2, 3, 4, 5, 6]),
    (7, [-3, 2, -1, 4, 5, -6, 8]),
    (1, [3])
]

# Execute testing using a loop
for test_case in test_cases:
    print(f"Testing with input: {test_case}")
    main(test_case)
    print()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\admin\Desktop\Labs\3 course\ЯПЗТ\10> python 10.py
Testing with input: (5, [2, 4, 6, 8, 10])
Size of B = 5, B = [2, 4, 6, 8, 10]

Testing with input: (5, [1, 3, 5, 7, 9])
Size of B = 0, B = []

Testing with input: (6, [1, 2, 3, 4, 5, 6])
Size of B = 3, B = [2, 4, 6]

Testing with input: (7, [-3, 2, -1, 4, 5, -6, 8])
Size of B = 4, B = [-6, 2, 4, 8]

Testing with input: (1, [3])
Size of B = 0, B = []
```

Висновок:

Отже, ця лабораторна робота з низхідного тестування дала глибоке розуміння підходу низхідного тестування та його практичної реалізації для конкретної задачі програмування. Основною метою було розробити програму, яка фільтрує та сортує парні числа з вхідного цілочисельного масиву, причому кінцевий програмний продукт складався з декількох модулів, включаючи введення початкових даних, фільтрацію парних чисел, сортування масиву та виведення результату.

Протягом цієї лабораторної роботи ми виконали кілька кроків для застосування методу низхідного тестування:

Ми створили головний модуль управління, функцію з іменем main, яка слугує оркестром потоку програми. Вона викликає інші функції, що відповідають за введення початкових даних, обробку даних та виведення результату.

Ми розробили заглушки для функцій input_data, filter_even_numbers, sort_even_numbers та output_result. Ці заглушки імітують функціональність відсутніх модулів, що дозволяє нам тестувати основний модуль та його взаємодію з іншими модулями.

Ми підготували тестові кейси, які охоплюють різні сценарії для вхідного масиву A. Тестові кейси включали ситуації, коли вхідний масив містив лише парні числа, лише непарні числа, суміш парних і непарних чисел, додатних і від'ємних чисел, а також одноелементний масив.

Тестування проводилося шляхом запуску основного модуля з підготовленими тестовими прикладами за допомогою розроблених заглушок. Ми порівняли результати роботи програми з очікуваними для кожного тесту.

Нарешті, ми представили результати, підготувавши звіт, що містить схему інтеграції, тексти заглушок, тестових кейсів та результати тестування у вигляді скріншотів.

Протягом цього процесу ми зіткнулися як з перевагами, так і з недоліками низхідного підходу до тестування. Основна перевага полягає в тому, що цей метод поєднує в собі тестування модулів, тестування зв'язків і частково тестування зовнішніх функцій. Інша перевага полягає в тому, що коли модулі вводу/виводу підключені, тести можуть бути підготовлені в зручній формі. Крім

того, підхід "зверху вниз" корисний, коли є сумніви щодо працездатності програми в цілому або коли є серйозні дефекти проектування.

Однак, метод низхідного тестування має деякі недоліки. Один з головних недоліків полягає в тому, що модулі рідко ретельно тестуються одразу після їх підключення. Інший незначний недолік полягає в тому, що він може породити віру в можливість почати програмування і тестування верхнього рівня програми до того, як вся програма буде повністю спроектована, що може призвести до опору поліпшенню структури програми.

Загалом, лабораторна робота дала цінне уявлення про методологію низхідного тестування, її переваги та недоліки, а також про її практичне застосування. Дотримуючись окреслених кроків та розуміючи важливість кожного етапу в процесі низхідного тестування, ми отримали всебічне розуміння цього підходу та успішно продемонстрували його застосування для заданої задачі програмування.