# Лабораторная работа № 5

## «Основные характеристики качества и надежности программного продукта»

***Задание 1. Постановка задачи***

Программа выполняет следующие операции над двумя матрицами:

1. Сложение матриц: Поэлементное сложение двух матриц одинаковой размерности.
2. Вычитание матриц: Поэлементное вычитание двух матриц одинаковой размерности.
3. Умножение матрицы на число: Умножает каждый элемент матрицы на заданное число, изменяя значения всех элементов, но сохраняя их расположение в матрице.
4. След матрицы: Сумма элементов главной диагонали (от верхнего левого до нижнего правого угла).

Программа имеет некоторые ограничения в виду специфики операций над матрицами:

* Для сложения и вычитания матрицы должны иметь одинаковые размеры.
* След можно вычислить только для квадратных матриц.
* Программа предполагает, что пользователь вводит корректные целые числа в качестве элементов матрицы.

***Задание 2. Теоретическая часть***

Тестирование программного обеспечения – это процесс проверки корректности работы программы с целью выявления ошибок и недочетов, а также обеспечения соответствия требованиям. Оно включает в себя выполнение программы с различными входными данными и сравнение фактических результатов с ожидаемыми. Тестирование помогает предотвратить ошибки и повысить надежность программного обеспечения.

Типы тестирования:

1. Модульное тестирование – проверка отдельных функций программы на соответствие требованиям.
2. Интеграционное тестирование – проверка взаимодействия нескольких модулей программы.
3. Системное тестирование – тестирование всей системы в целом для проверки её работоспособности.
4. Регрессионное тестирование – повторное тестирование программы после внесения изменений, чтобы убедиться в отсутствии новых ошибок.

Уровни тестирования:

* Функциональное тестирование – проверяет, выполняет ли программа требуемые операции корректно.
* Нефункциональное тестирование – оценивает производительность, удобство использования и надежность программы.

Отладка – это процесс выявления, анализа и исправления ошибок в программном обеспечении. Он включает в себя:

* Воспроизведение ошибки: Проверка того, что ошибка возникает стабильно при определенных условиях.
* Выявление первопричины: Анализ кода и определение участка, вызывающего некорректное поведение.
* Исправление ошибки: Изменение кода для устранения проблемы.
* Проверка исправления: Повторное тестирование, чтобы убедиться, что ошибка устранена, а новые проблемы не появились.

К методам отладки относится:

1. Отладка с помощью вывода: Использование операторов вывода для анализа промежуточных значений в программе.
2. Использование средств отладки: Работа с отладчиком в IDE для пошагового выполнения кода и отслеживания значений переменных.
3. Код-ревью: Проверка кода другим разработчиком для поиска возможных ошибок и багов.
4. Модульное тестирование: Создание тестов для проверки корректности отдельных частей программы.

***Задание 3. Практическая часть***

Следующие наборы тестов, описанные в таблице 1, предназначены для тестирования матричных операций, реализованных в программе. Эти тестовые примеры охватывают различные сценарии, включая допустимые входные данные, крайние случаи и условия ошибок.

Таблица 1 – Тестовые кейсы для кода, выполняющего операции над матрицами

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый пример | Ввод данных | Ожидаемый результат | Фактический результат |
| Корректность генерации матрицы | Генерация матриц  Столбцы = 2  Колонки = 2 | Сгенерированные матрицы размером 2\*2 | Матрица 1:  [3, 7]  [5, 2]  Матрица 2:  [6, 1]  [4, 8] |

Продолжение таблицы 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Корректность ввода матрицы с клавиатуры | Ввод с клавиатуры  Столбцы = 3  Колонки = 3 | Матрицы, введенные вручную | Матрица 1:  [1, 2, 3]  [4, 5, 6]  [7, 8, 9]  Матрица 2:  [9, 8, 7]  [6, 5, 4]  [3, 2, 1] |
| Обработка нецелочисленных значений | Ввод "abc" | Ошибка, указывающая на некорректность данных | Ошибка: Введите целое число. |
| Сложение матриц | Генерация матриц  Столбцы = 2  Колонки = 2 | Матрица, являющаяся суммой | Сложение матриц:  [9, 8]  [9, 10] |
| Вычитание матриц | Генерация матриц  Столбцы = 2  Колонки = 2 | Матрица, являющаяся разницей | Вычитание матриц:  [-3, 6]  [1, -6] |
| Умножение матрицы на число | Генерация матриц  Столбцы = 2  Колонки = 2 | Матрица, умноженная на 3 | Матрица 1, умноженная на число:  [9, 21]  [15, 6]  Матрица 2, умноженная на число:  [18, 3]  [12, 24] |
| След матрицы | Генерация матриц  Столбцы = 2  Колонки = 2 | След матрицы (сумма диагональных элементов) | След матрицы 1:  5  След матрицы 2:  14 |
| Обработка матрицы 1\*1 | Генерация матриц  Столбцы = 1  Колонки = 1 | Корректные операции над матрицами | Матрица 1:  [4]  Матрица 2:  [5]  Сложение матриц:  [9]  Вычитание матриц:  [-1]  Матрица 1, умноженная на число (3):  [12] |

Продолжение таблицы 1.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Матрица 2, умноженная на число (3):  [15]  След матрицы 1:  4  След матрицы 2:  5 |
| Обработка матрицы 50\*50 | Генерация матриц  Столбцы = 50  Колонки = 50 | Корректные операции над матрицами | Программа корректно обрабатывает матрицу большого размера |
| Обработка неположительных значений | Генерация матриц  Столбцы = 0  Колонки = 0 | Ошибка, указывающая на некорректность введенных данных | Ошибка: Введите положительное целое число. |
| Обработка недопустимого варианта меню | Ввод варианта, отличный от 1 или 2 | Программа должна предложить пользователю ввести правильный вариант | Ошибка: Введите 1 или 2. |

Сформированные тестовые наборы охватывают различные сценарии, что позволяет полноценно проверить работу матричных операций. Запуск тестов позволяет удостовериться, что программа корректно выполняет задачи при различных входных данных, включая стандартные, крайние случаи и ошибочные данные. Такой комплексный подход к тестированию обеспечивает стабильность и надежность программного обеспечения.

# Приложение А

import random

def generate\_random\_matrix(rows, cols):

"""Генерация матрицы случайными числами от 1 до 10"""

return [[random.randint(1, 10) for \_ in range(cols)] for \_ in range(rows)]

def print\_matrix(matrix):

"""Вывод матрицы в консоль"""

for row in matrix:

print(row)

def matrix\_addition(matrix1, matrix2):

"""Сложение матриц"""

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):

print("Ошибка: Размеры матриц не совпадают для сложения.")

return None

return [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]

def matrix\_subtraction(matrix1, matrix2):

"""Вычитание матриц"""

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):

print("Ошибка: Размеры матриц не совпадают для вычитания.")

return None

return [[matrix1[i][j] - matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(matrix1))]

def multiply\_matrix\_by\_number(matrix, number):

"""Умножение матрицы на число"""

return [[matrix[i][j] \* number for j in range(len(matrix[0]))] for i in range(len(matrix))]

def matrix\_trace(matrix):

"""Вычисление следа матрицы (сумма элементов главной диагонали)"""

if len(matrix) != len(matrix[0]):

print("Ошибка: След можно вычислить только у квадратной матрицы.")

return None

return sum(matrix[i][i] for i in range(len(matrix)))

def get\_positive\_integer(prompt):

"""Запрос положительного целого числа у пользователя"""

while True:

try:

value = int(input(prompt))

if value > 0:

return value

else:

print("Ошибка: Введите положительное целое число.")

except ValueError:

print("Ошибка: Введите целое число.")

def get\_integer(prompt):

"""Запрос целого числа у пользователя"""

while True:

try:

return int(input(prompt))

except ValueError:

print("Ошибка: Введите целое число.")

def input\_matrix(rows, cols):

"""Ввод матрицы пользователем"""

matrix = []

for i in range(rows):

row = []

for j in range(cols):

row.append(get\_integer(f"({i + 1},{j + 1}): "))

matrix.append(row)

return matrix

def main():

print("1. Заполнить матрицы случайными числами")

print("2. Ввести матрицы вручную")

while True:

choice = get\_integer("Введите ваш выбор (1 или 2): ")

if choice in [1, 2]:

break

print("Ошибка: Введите 1 или 2.")

print("\nВведите размеры первой матрицы:")

rows1 = get\_positive\_integer("Количество строк: ")

cols1 = get\_positive\_integer("Количество столбцов: ")

print("\nВведите размеры второй матрицы:")

rows2 = get\_positive\_integer("Количество строк: ")

cols2 = get\_positive\_integer("Количество столбцов: ")

if choice == 1:

matrix1 = generate\_random\_matrix(rows1, cols1)

matrix2 = generate\_random\_matrix(rows2, cols2)

else:

print("\nМатрица 1:")

matrix1 = input\_matrix(rows1, cols1)

print("\nМатрица 2:")

matrix2 = input\_matrix(rows2, cols2)

print("\nМатрица 1:")

print\_matrix(matrix1)

print("\nМатрица 2:")

print\_matrix(matrix2)

print("\nСложение матриц:")

result = matrix\_addition(matrix1, matrix2)

if result:

print\_matrix(result)

print("\nВычитание матриц:")

result = matrix\_subtraction(matrix1, matrix2)

if result:

print\_matrix(result)

num\_1 = get\_integer("\nВведите число для умножения первой матрицы: ")

num\_2 = get\_integer("\nВведите число для умножения второй матрицы: ")

print("\nМатрица 1, умноженная на число:")

print\_matrix(multiply\_matrix\_by\_number(matrix1, num\_1))

print("\nМатрица 2, умноженная на число:")

print\_matrix(multiply\_matrix\_by\_number(matrix2, num\_2))

if rows1 == cols1:

print("\nСлед первой матрицы:")

trace = matrix\_trace(matrix1)

if trace is not None:

print(trace)

if rows2 == cols2:

print("\nСлед второй матрицы:")

trace = matrix\_trace(matrix2)

if trace is not None:

print(trace)

if \_\_name\_\_ == "\_\_main\_\_":

main()