Отчёт  
по лабораторной работе № 11

по дисциплине: «МДК 02.02 Инструментальные средства разработки программного обеспечения»

Тема: «Разработка структуры проекта»

|  |  |  |
| --- | --- | --- |
| Студент гр. 22 ИТ |  | Липинский К.С. |
| Преподаватель |  | Быковский З.С. |

### Цель работы.

Формирование навыков постановки задачи и разработки технического задания на программный продукт.

### Выполнение работы.

Задание 1.

Техническое задание (далее – ТЗ):

**Введение**

Программа, предназначенная для вычисления суммы столбцов матрицы размером n×m, содержащих хотя бы один нулевой элемент, разработана на языке C#. Она обеспечивает удобный и интуитивно понятный интерфейс для пользователя.

**Основание для разработки**

Целью создания данной программы является предоставление пользователю инструмента для быстрого и эффективного решения задачи. Программа позволяет вводить данные через консоль и получать результаты в понятной форме.

**Назначение**

Программа предназначена для выполнения задачи по нахождению суммы столбцов матрицы, содержащих хотя бы один нулевой элемент. Она обеспечивает простой и удобный интерфейс, что делает её использование максимально комфортным для пользователя.

**Требования к программе и программному продукту**

*Функциональные требования*

* Программа должна выполнять только поставленные задачи.
* Программа должна быть безопасной для использования.

*Нефункциональные требования*

Программа должна иметь простой и интуитивно понятный интерфейс для пользователя.

*Требования к программной документации*

Программная документация должна включать:

* Правила использования программы.
* Руководство по устранению или предотвращению ошибок.
* Описание возможных ошибок.
* Описание назначения программы.

Код:

import random  
  
n = int(input("n: "))  
m = int(input("m: "))  
  
matrix = []  
  
for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
  
for row in range(n):  
 for column in range(m):  
 print(matrix[row][column], end=" ")  
 print("")

Результат:



Задание 2.

ТЗ:

**Введение**

Программа, предназначенная для вычисления кол-ва положительных элементов, содержащихся в матрице n \* m, если a ij = sin(i+j/2), разработана на языке C#. Она обеспечивает удобный и интуитивно понятный интерфейс для пользователя.

**Основание для разработки**

Целью создания данной программы является предоставление пользователю инструмента для быстрого и эффективного решения задачи. Программа позволяет вводить данные через консоль и получать результаты в понятной форме.

**Назначение**

Программа предназначена для выполнения задачи по вычислению кол-ва положительных элементов, содержащихся в матрице n \* m, если a ij = sin(i+j/2). Она обеспечивает простой и удобный интерфейс, что делает её использование максимально комфортным для пользователя.

**Требования к программе и программному продукту**

*Функциональные требования*

* Программа должна выполнять только поставленные задачи.
* Программа должна быть безопасной для использования.

*Нефункциональные требования*

Программа должна иметь простой и интуитивно понятный интерфейс для пользователя.

*Требования к программной документации*

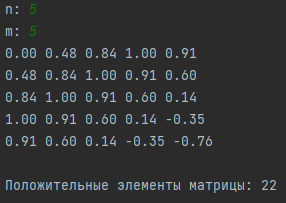
Программная документация должна включать:

* Правила использования программы.
* Руководство по устранению или предотвращению ошибок.
* Описание возможных ошибок.
* Описание назначения программы.

Код:

import math  
  
n = int(input("n: "))  
m = int(input("m: "))  
  
matrix = []  
positive = 1  
  
for i in range(n):  
 row = []  
 for j in range(m):  
 value = math.sin((i + j) / 2)  
 row.append(value)  
 if value > 0:  
 positive += 1  
 matrix.append(row)  
  
for row in matrix:  
 for value in row:  
 print(f"{value:.2f}", end=" ")  
 print("")  
  
print(f"\nПоложительные элементы матрицы: {positive}")=

Результат:



Задание 3.

ТЗ:

**Введение**

Программа, предназначенная для вычисления симметричности матрицы относительно главной диагонали, разработана на языке C#. Она обеспечивает удобный и интуитивно понятный интерфейс для пользователя.

**Основание для разработки**

Целью создания данной программы является предоставление пользователю инструмента для быстрого и эффективного решения задачи. Программа позволяет вводить данные через консоль и получать результаты в понятной форме.

**Назначение**

Программа предназначена для выполнения задачи по вычислению симметричности матрицы относительно главной диагонали. Она обеспечивает простой и удобный интерфейс, что делает её использование максимально комфортным для пользователя.

**Требования к программе и программному продукту**

*Функциональные требования*

* Программа должна выполнять только поставленные задачи.
* Программа должна быть безопасной для использования.

*Нефункциональные требования*

Программа должна иметь простой и интуитивно понятный интерфейс для пользователя.

*Требования к программной документации*

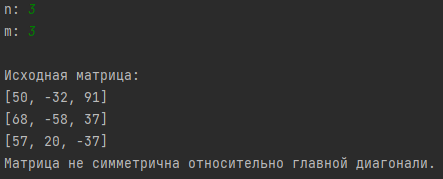
Программная документация должна включать:

* Правила использования программы.
* Руководство по устранению или предотвращению ошибок.
* Описание возможных ошибок.
* Описание назначения программы.

Код:

import random  
  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
  
 return n, m  
  
  
def fill\_matrix():  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
  
  
def is\_symmetric(matrix):  
 n = len(matrix)  
 for i in range(n):  
 for j in range(n):  
 if matrix[i][j] != matrix[j][i]:  
 return False  
 return True  
  
  
matrix = []  
n, m = input\_size()  
fill\_matrix()  
  
print("\nИсходная матрица:")  
for row in matrix:  
 print(row)  
  
if is\_symmetric(matrix):  
 print("Матрица симметрична относительно главной диагонали.")  
else:  
 print("Матрица не симметрична относительно главной диагонали.")

Результат:



Задание 4.

ТЗ:

**Введение**

Программа, предназначенная для транспонирования квадратичной вещественной матрицы размерности n×m, разработана на языке C#. Она обеспечивает удобный и интуитивно понятный интерфейс для пользователя.

**Основание для разработки**

Целью создания данной программы является предоставление пользователю инструмента для быстрого и эффективного решения задачи. Программа позволяет вводить данные через консоль и получать результаты в понятной форме.

**Назначение**

Программа предназначена для выполнения задачи по транспонированию квадратичной вещественной матрицы размерности n×m. Она обеспечивает простой и удобный интерфейс, что делает её использование максимально комфортным для пользователя.

**Требования к программе и программному продукту**

*Функциональные требования*

* Программа должна выполнять только поставленные задачи.
* Программа должна быть безопасной для использования.

*Нефункциональные требования*

Программа должна иметь простой и интуитивно понятный интерфейс для пользователя.

*Требования к программной документации*

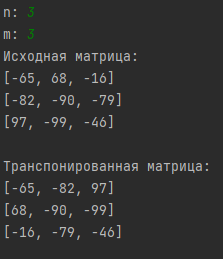
Программная документация должна включать:

* Правила использования программы.
* Руководство по устранению или предотвращению ошибок.
* Описание возможных ошибок.
* Описание назначения программы.

Код:

import random  
  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
  
 if n == m:  
 return n, m  
 else:  
 print("Матрица должна быть квадратной. Попробуйте снова.")  
  
  
def fill\_matrix():  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
  
  
def transpose\_matrix():  
 n = len(matrix)  
 m = len(matrix[0])  
  
 transposed = [[0] \* n for \_ in range(m)]  
  
 for i in range(n):  
 for j in range(m):  
 transposed[j][i] = matrix[i][j]  
  
 return transposed  
  
matrix = []  
n, m = input\_size()  
fill\_matrix()  
transposed\_matrix = transpose\_matrix()  
  
print("Исходная матрица:")  
for row in matrix:  
 print(row)  
  
print("\nТранспонированная матрица:")  
for row in transposed\_matrix:  
 print(row)

Результат:

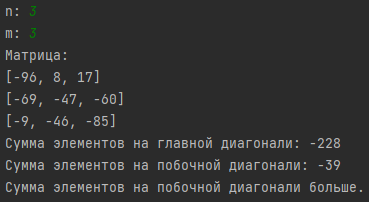


Задание 5.

Код:

import random  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
 return n, m  
  
def fill\_matrix(n, m):  
 matrix = []  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
 return matrix  
  
def check\_sum\_diagonals(matrix):  
 n = len(matrix)  
 main\_diagonal\_sum = 0  
 secondary\_diagonal\_sum = 0  
  
 for i in range(n):  
 if i < len(matrix[i]):  
 main\_diagonal\_sum += matrix[i][i]  
 if i < len(matrix[i]):  
 secondary\_diagonal\_sum += matrix[i][n - i - 1]  
  
 return main\_diagonal\_sum, secondary\_diagonal\_sum  
  
matrix = []  
n, m = input\_size()  
matrix = fill\_matrix(n, m)  
  
# Вывод матрицы  
print("Матрица:")  
for row in matrix:  
 print(row)  
  
main\_sum, secondary\_sum = check\_sum\_diagonals(matrix)  
print(f"Сумма элементов на главной диагонали: {main\_sum}")  
print(f"Сумма элементов на побочной диагонали: {secondary\_sum}")  
  
if main\_sum > secondary\_sum:  
 print("Сумма элементов на главной диагонали больше.")  
elif main\_sum < secondary\_sum:  
 print("Сумма элементов на побочной диагонали больше.")  
else:  
 print("Суммы элементов на главной и побочной диагоналях равны.")

Результат:

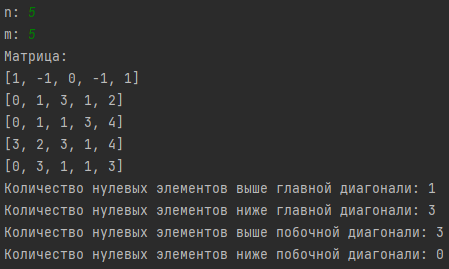


Задание 6.

Код:

import random  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
 return n, m  
  
def fill\_matrix(n, m):  
 matrix = []  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-1, 5, 1))  
 matrix.append(a)  
 return matrix  
  
def count\_zeros(matrix):  
 n = len(matrix)  
 above\_main\_diagonal = 0  
 below\_main\_diagonal = 0  
 above\_secondary\_diagonal = 0  
 below\_secondary\_diagonal = 0  
  
 for i in range(n):  
 for j in range(n):  
 if i < j and matrix[i][j] == 0:  
 above\_main\_diagonal += 1  
 elif i > j and matrix[i][j] == 0:  
 below\_main\_diagonal += 1  
 if i < (n - j - 1) and matrix[i][j] == 0:  
 above\_secondary\_diagonal += 1  
 elif i > (n - j - 1) and matrix[i][j] == 0:  
 below\_secondary\_diagonal += 1  
  
 return above\_main\_diagonal, below\_main\_diagonal, above\_secondary\_diagonal, below\_secondary\_diagonal  
  
matrix = []  
n, m = input\_size()  
matrix = fill\_matrix(n, m)  
  
# Вывод матрицы  
print("Матрица:")  
for row in matrix:  
 print(row)  
  
above\_main, below\_main, above\_secondary, below\_secondary = count\_zeros(matrix)  
print(f"Количество нулевых элементов выше главной диагонали: {above\_main}")  
print(f"Количество нулевых элементов ниже главной диагонали: {below\_main}")  
print(f"Количество нулевых элементов выше побочной диагонали: {above\_secondary}")  
print(f"Количество нулевых элементов ниже побочной диагонали: {below\_secondary}")

Результат:

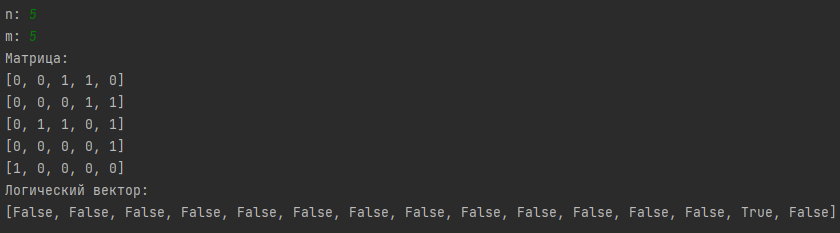


Задание 7.

Код:

import random  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
 return n, m  
  
def fill\_matrix(n, m):  
 matrix = []  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(0, 2, 1))  
 matrix.append(a)  
 return matrix  
  
def get\_logical\_vector(matrix):  
 n = len(matrix)  
 m = len(matrix[0])  
 logical\_vector = []  
  
 for k in range(m):  
 # Условие: все элементы k столбца нулевые  
 all\_zeros\_in\_column = all(matrix[i][k] == 0 for i in range(n))  
 logical\_vector.append(all\_zeros\_in\_column)  
  
 for k in range(n):  
 # Условие: элементы k строки матрицы упорядочены по убыванию  
 sorted\_descending = all(matrix[k][j] >= matrix[k][j + 1] for j in range(m - 1))  
 logical\_vector.append(sorted\_descending)  
  
 # Условие: k строка массива симметрична  
 symmetric\_row = matrix[k] == matrix[k][::-1]  
 logical\_vector.append(symmetric\_row)  
  
 return logical\_vector  
  
matrix = []  
n, m = input\_size()  
matrix = fill\_matrix(n, m)  
  
# Вывод матрицы  
print("Матрица:")  
for row in matrix:  
 print(row)  
  
logical\_vector = get\_logical\_vector(matrix)  
print("Логический вектор:")  
print(logical\_vector)

Результат:

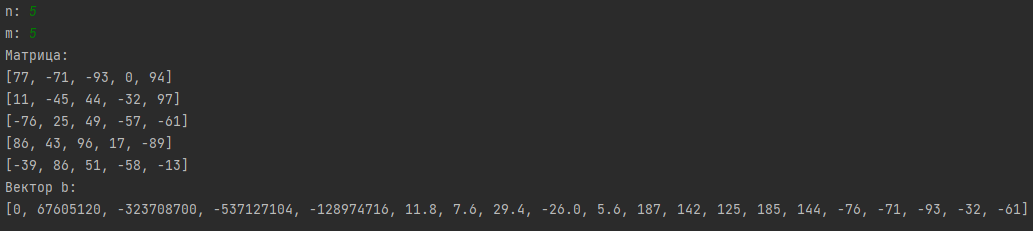


Задание 8.

Код:

import random  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
 return n, m  
  
def fill\_matrix(n, m):  
 matrix = []  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
 return matrix  
  
def form\_vector\_b(matrix):  
 n = len(matrix)  
 m = len(matrix[0])  
 vector\_b = []  
  
 # Произведение элементов соответствующих строк  
 for row in matrix:  
 product = 1  
 for element in row:  
 product \*= element  
 vector\_b.append(product)  
  
 # Среднее арифметическое соответствующих столбцов  
 for col in range(m):  
 column\_sum = sum(matrix[row][col] for row in range(n))  
 column\_avg = column\_sum / n  
 vector\_b.append(column\_avg)  
  
 # Разность наибольших и наименьших элементов соответствующих строк  
 for row in matrix:  
 max\_element = max(row)  
 min\_element = min(row)  
 difference = max\_element - min\_element  
 vector\_b.append(difference)  
  
 # Значения первых отрицательных элементов в столбце  
 for col in range(m):  
 first\_negative = None  
 for row in range(n):  
 if matrix[row][col] < 0:  
 first\_negative = matrix[row][col]  
 break  
 if first\_negative is not None:  
 vector\_b.append(first\_negative)  
 else:  
 vector\_b.append(0) # Если отрицательных элементов нет, добавляем 0  
  
 return vector\_b  
  
matrix = []  
n, m = input\_size()  
matrix = fill\_matrix(n, m)  
  
# Вывод матрицы  
print("Матрица:")  
for row in matrix:  
 print(row)  
  
vector\_b = form\_vector\_b(matrix)  
print("Вектор b:")  
print(vector\_b)

Результат:

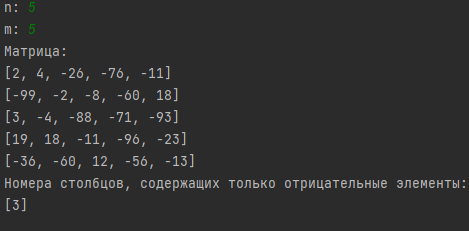


Задание 9.

Код:

import random  
  
def input\_size():  
 while True:  
 n = int(input("n: "))  
 m = int(input("m: "))  
 return n, m  
  
def fill\_matrix(n, m):  
 matrix = []  
 for row in range(n):  
 a = []  
 for column in range(m):  
 a.append(random.randrange(-99, 99, 1))  
 matrix.append(a)  
 return matrix  
  
def find\_negative\_columns(matrix):  
 m = len(matrix[0])  
 negative\_columns = []  
  
 for col in range(m):  
 all\_negative = all(matrix[row][col] < 0 for row in range(len(matrix)))  
 if all\_negative:  
 negative\_columns.append(col)  
  
 return negative\_columns  
  
matrix = []  
n, m = input\_size()  
matrix = fill\_matrix(n, m)  
  
# Вывод матрицы  
print("Матрица:")  
for row in matrix:  
 print(row)  
  
negative\_columns = find\_negative\_columns(matrix)  
print("Номера столбцов, содержащих только отрицательные элементы:")  
print(negative\_columns)

Результат:



### Контрольные вопросы.

1. Перечислите этапы разработки программных продуктов.

**Анализ требований:** Сбор и анализ требований к системе, определение целей и задач.

**Проектирование:** Разработка архитектуры системы, создание моделей данных и интерфейсов.

**Разработка:** Написание кода, создание компонентов системы.

**Тестирование:** Проверка работоспособности системы, выявление и исправление ошибок.

**Внедрение:** Установка и настройка системы на рабочем месте, обучение пользователей.

**Эксплуатация и сопровождение:** Мониторинг работы системы, техническая поддержка, обновления и доработки.

1. Для чего необходимо техническое задание?

**Техническое задание (ТЗ)** необходимо для четкого определения требований к разрабатываемому программному продукту. Оно служит основой для планирования работ, оценки ресурсов и сроков, а также для контроля качества на всех этапах разработки. ТЗ помогает избежать недоразумений между заказчиком и исполнителем, обеспечивая единое понимание конечного результата.

1. Кто занимается разработкой технического задания?

**Разработкой технического задания** обычно занимаются системные аналитики, бизнес-аналитики или специалисты по технической документации. В некоторых случаях в процесс могут быть вовлечены представители заказчика, технические специалисты и менеджеры проекта.

1. Какие пункты включает техническое задание?

Техническое задание обычно включает следующие пункты:

**Введение:** Общая информация о проекте, цели и задачи.

**Описание системы:** Функциональные и нефункциональные требования, архитектура системы.

**Требования к интерфейсам:** Описание пользовательских и программных интерфейсов.

**Требования к производительности:** Ожидаемая нагрузка, время отклика, пропускная способность.

**Требования к безопасности:** Меры по защите данных, аутентификация и авторизация.

**Требования к совместимости:** Поддержка различных операционных систем, браузеров, устройств.

**Требования к документации:** Описание необходимых видов документации (пользовательская, техническая и т.д.).

**План работ:** Этапы разработки, сроки выполнения, ресурсы.

**Критерии приемки:** Условия, при которых система считается завершенной и готовой к эксплуатации.