МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»»

ИНЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентированный анализ и программирование»

Тема: «Особенности реализации принципа наследования в языке Python на примере конкретной задачи»

Студент группы: ИЭозс-62-22 Муратов А.Д..
(Ф.И.О.)
Руководитель: Хацкевич А.А.
(уч. степень, звание, Ф.И.О.)

Москва 2023

Оглавление

Введение ... 3

1. Объектно-ориентированный анализ ... 5

1.1. Диаграмма UML – диаграмма классов ... 5

1.2. Диаграмма UML – диаграмма последовательности ... 10

1.3. Диаграмма UML – диаграмма деятельности ... 12

2. Объектно-ориентированное программирование ... 15

2.1. Разработка консольного приложения с использованием подпрограмм ... 15

2.1.1. Описание функций ... 15

2.1.2. Состав данных консольного приложения с использованием подпрограмм ... 16

2.1.3. Шаблон функций и форма ввода-вывода ... 17

2.1.4. Блок-схема алгоритма функций ... 17

2.1.5. Программный код консольного приложения с использованием подпрограмм. ... 19

2.1.6. Тестирование программы консольного приложения с использованием подпрограмм. ... 20

2.2. Разработка консольного приложения с использованием классов ... 21

2.2.1. Описание класса ... 21

2.2.2. Состав данных консольного приложения с использованием классов ... 22

2.2.3. Блок-схема консольного приложения с использованием классов. ... 23

2.2.4. Программный код консольного приложения с использованием классов. ... 25

2.2.7. Тестирование программы консольного приложения с использованием подпрограмм. ... 27

2.3. Разработка оконного приложения ... 28

2.3.1. Процесс разработки оконного приложения ... 28

2.3.2. Тестирование программы оконного приложения ... 42

Заключение ... 45

Список литературы ... 47

Введение

Объектно-ориентированное программирование (ООП) — совокупность принципов, технологии и инструментальных средств для создания программных систем, в основу которых закладывается архитектура взаимодействия объектов.

Объектно-ориентированное программирование сильно упрощает процесс организации и создания структуры программы. Отдельные объекты, которые можно менять без воздействия на остальные части программы, упрощают также и внесение в программу изменений. Так как с течением времени программы становятся всё более крупными, а их поддержка всё более тяжёлой, эти два аспекта ООП становятся всё более актуальными.

Python — это высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений, web, парсинг и другие.

Python смог захватить малую часть рынка веб-разработки, иногда используется для написания десктопных приложений и, конечно, тотально доминирует в сфере машинного обучения. Кроме того, на нём создаётся много прототипов, которые позволяют быстро набросать функционал и внешний вид будущего проекта.

Цель объектно-ориентированного программирования состоит в повторном использовании созданных вами классов, что экономит ваше время и силы. Если вы уже создали некоторый класс, то возможны ситуации, что новому классу нужны многие или даже все особенности уже существующего класса, и необходимо добавить один или несколько элементов данных или функций. Другими словами, новый объект будет наследовать элементы существующего класса (называемого базовым классом). Когда вы строите новый класс из существующего, этот новый класс часто называется производным классом.

Главная цель объектно-ориентированного программирования (ООП) — это упрощение работы с кодом путем оптимизации и следовательно

сокращения количества написанного текста, что повышает читаемость кода, адаптивность и как следствие работоспособность и полезность. В Python благодаря классам и принципу наследования (а также «фишкам» языка), которым они обладают, весьма объёмную программу, зачастую, можно сократить в несколько раз.

Объектно-ориентированный анализ (ООА) направлен на создание моделей, более близких к реальности, с использованием объектно-ориентированного подхода; это методология, при которой требования формируются на основе понятий классов и объектов, составляющих словарь предметной области.

На результатах ООА формируются модели, на которых основывается объектно-ориентированное проектирование; объектно-ориентированное проектирование в свою очередь создает основу для окончательной реализации системы с использованием методологии объектно-ориентированного программирования.

1. Объектно-ориентированный анализ

1.1. Диаграмма UML – диаграмма классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Название диаграммы: «учет посещаемости и успеваемости студентов колледжа».

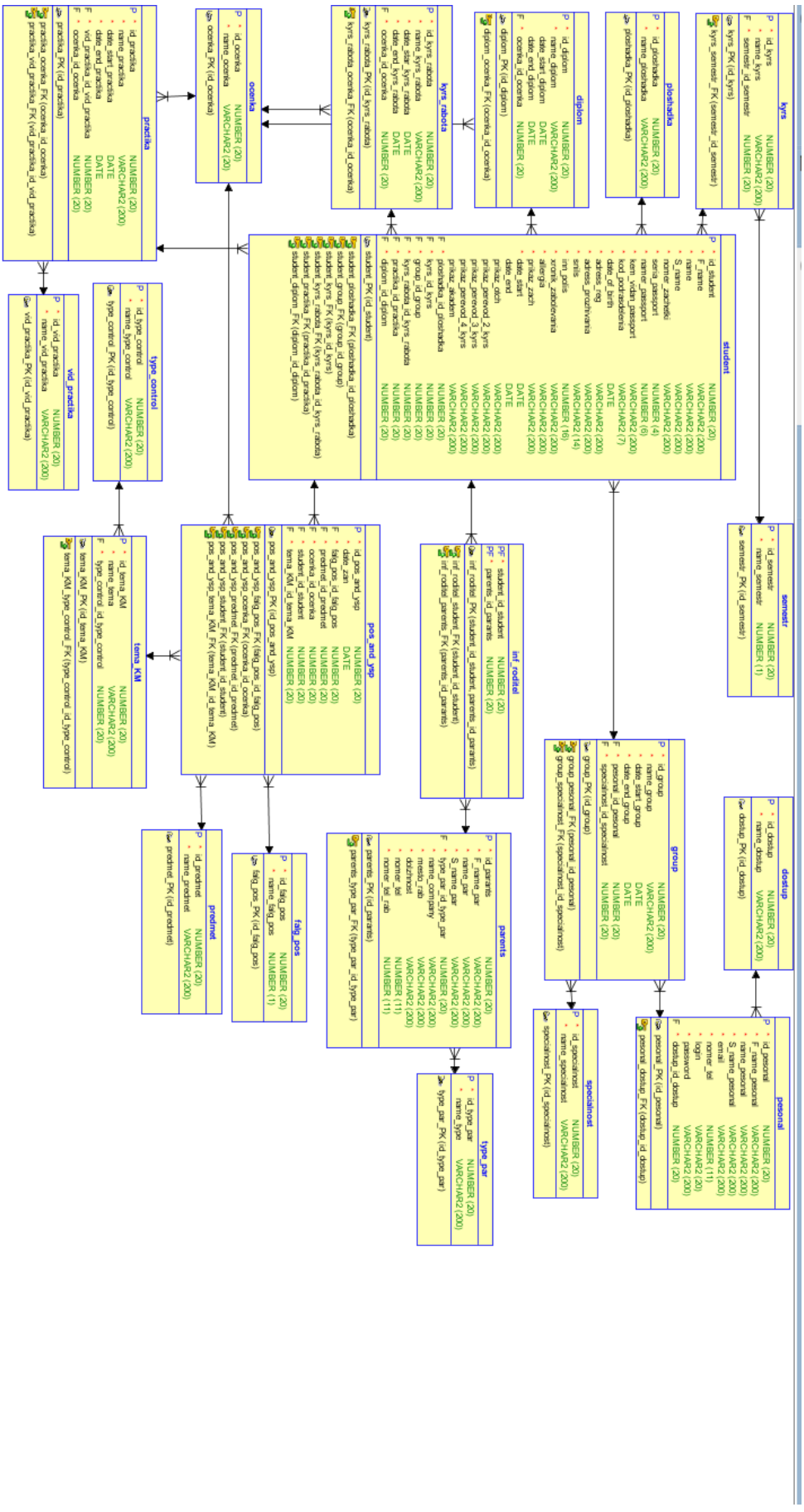

Рисунок 1.1. Диаграмма классов учет посещаемости и успеваемости студентов колледжа

Описание и анализ диаграммы классов

Данная диаграмма (рисунок 1.1.) классов описывает типы объектов системы и отношения, которые существуют между ними. Всего в данной системе 21 класс.

Сущности и их атрибуты:

Сущность kyrs и ее атрибуты:

- id_kyrs: number(20)
- name_kyrs: varchar2(200)
- semester_id_semestr: number(20)

Сущность semestr и ее атрибуты:

- id_semestr: number(20)
- name_semestr: number(1)

Сущность ploshadka:

- id_ploshadka: number(20)
- name_ploshadka: varchar2(200)

Сущность diplom и ее атрибуты:

- id_diplom: number(20)
- name_diplom: varchar2(200)
- date_start_diplom: date
- date_end_diplom: date
- ocenka_id_ocenka: number(20)

Сущность ocenka и ее атрибуты:

- id_ocenka: namber(20)
- name_ocenka: varchar2(20)

Сущность kyrs_rabota и ее атрибуты:

- id_kyrs_rabota: varchar2(200)
- date_start_kyrs_rabota: date
- date_end_kyrs_rabota: date
- vid_practika_id_vid_practika: number(20)

- ocenka_id_ocenka: number(20)

Сущность practika и ее атрибуты:

- id_practika: number(20)
- name_practika: varchar2(200)
- date_start_practika: date
- date_end_practika: date
- ocenka_id_ocenka: number(20)

Сущность type_control и ее атрибуты:

- id_type_control: namber(20)
- name_type_control: varchar2(200)

Сущность vid_practika и ее атрибуты:

- id_vid_practika: number(20)
- name_vid_practika: varchar2(200)

Сущность tema_KM и ее атрибуты:

- id_tema_KM: number(20)
- name_tema: varchar2(200)
- type_control_id_type_control: number(20)

Сущность pos_and_ysp и ее атрибуты:

- id_pos_and_ysp: number(20)
- date_zan: date
- flag_pos_id_flag_pos: number(20)
- predmet_id_predmet: number(20)
- ocenka_id_ocenka: number(20)
- student_id_student: number(20)
- tema_KM_id_tema_KM: number(20)

Сущность flag_pos и ее атрибуты:

- id_flag_pos: number(20)
- name_flag_pos: number(1)

Сущность predmet и ее атрибуты:

- id_predmet: number(20)
- name_predmet: varchar2(200)

Сущность type_par и ее атрибуты:

- id_type_par: number(20)
- name_type: varchar2(200)

Сущность specialnost и ее атрибуты:

- id_specialnost: number(20)
- name_specialnost: varchar2(200)

Сущность parents и ее атрибуты:

- id_parants: number(20)
- F_name_par: varchar2(200)
- name_par: varchar2(200)
- S_name_parr: varchar2(200)
- type_par_id_type_par: number(20)
- name_company: varchar2(200)
- mesto_rab: varchar2(200)
- dolzhnost: varchar2(200)
- nomer_tel: number(11)
- nomer_tel_rab: number(11)

Сущность inf_roditel и ее атрибуты:

- student_id_student: number(20)
- parents_id_parants: number(20)

Сущность group и ее атрибуты:

- id_group: number(20)
- name_group: varchar2(200)
- date_start_group: date
- date_end_group: date
- personal_id_personal: number(20)
- specialnost_id_specialnost: number(20)

Сущность personal и ее атрибуты:

- id_personal: number(20)
- F_name_personal: varchar2(200)
- name_personal: varchar2(200)
- S_name_personal: varchar2(200)
- email: varchar2(200)
- nomer_tel: number(11)
- login: varchar2(20)
- password: varchar2(200)
- dustup_id_distup: number(20)

Сущность dostup и ее атрибуты:

- id_dostup: number(20)
- name_dostup: varchar2(200)

В данном разделе описана диаграмма классов для БД учета посещаемости и успеваемости студентов колледжа. Связи между сущностями: один ко многому, более подробно Вы можете ознакомится на рисунке 1.1.

1.2. Диаграмма UML – диаграмма последовательности

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Название диаграммы: Процесс оформление заказа на сайте

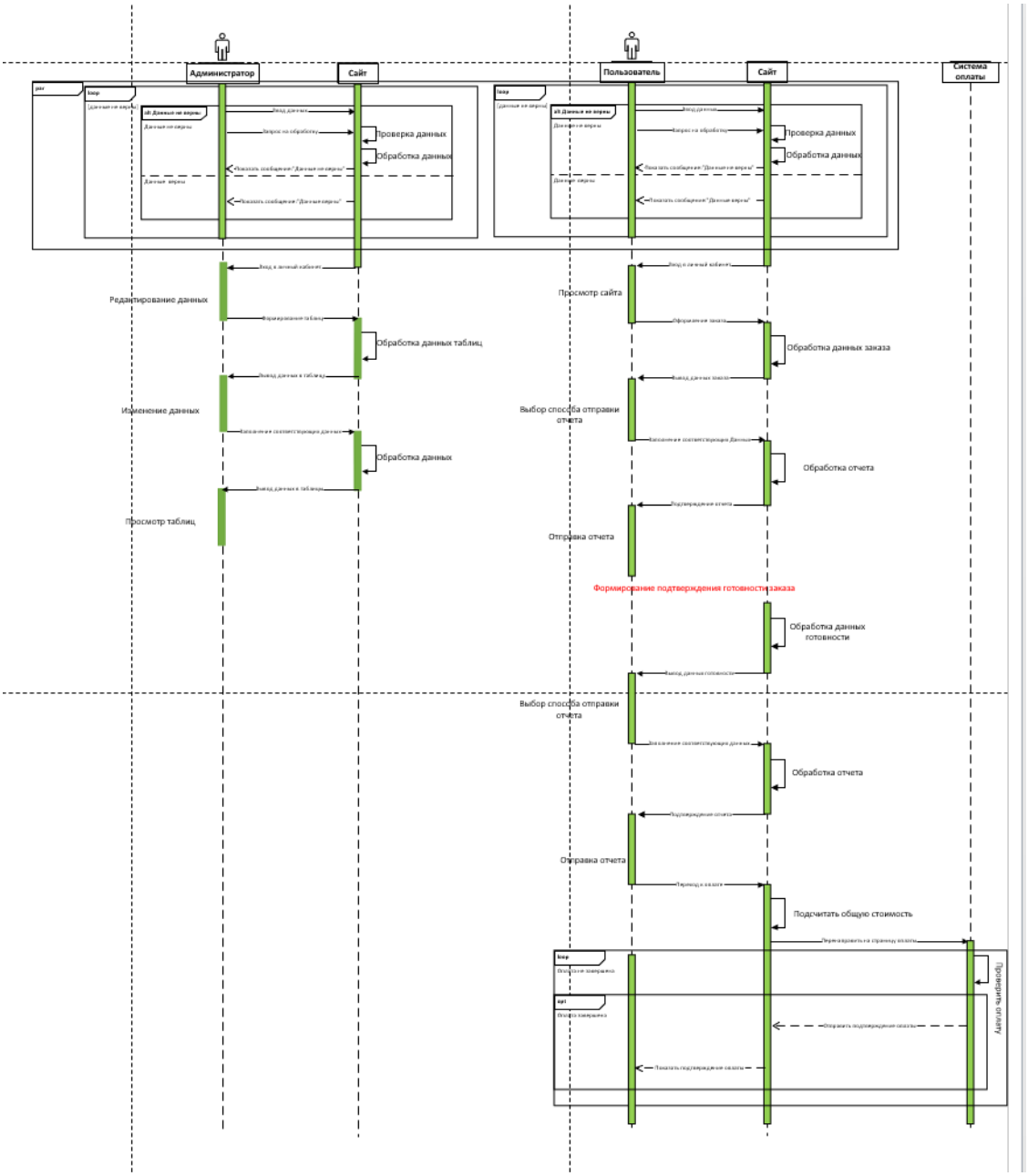

Рисунок 1.2. Диаграмма последовательностей процесса оформления заказа на сайте

Описание и анализ диаграммы последовательностей

На рисунке 1.2. показана диаграмма последовательности, сценария выполнения задачи, в нашем случае процесс оформления заказа в магазине. В данной диаграмме указано два актера – покупатель и администратор, а также добавлен класс «Сайт» и «Система оплаты». Клиент заходит на сайт для оформления заказа. Для его оформления, покупатель авторизуется на сайте, выбирает услугу из списка и выбирает способ ее получения, после чего

переходит на страницу оплаты. Администратор же, для работы, проходит авторизацию и может добавлять данные, а также вносить изменения в уже имеющиеся.

В итоге можно прийти к такому выводу что описание письменно определенного действия или сценария на порядок сложнее чем создать диаграмму со всеми возможными исходами событий, которые идут последовательно друг за другом со своими определенными жизненными циклами при этом контролироваться операторами «BREAK», «LOOP», «OPT», «ALT» и «PAR».

1.3. Диаграмма UML – диаграмма деятельности

Диаграмма для демонстрации рабочего процесса некоторой деятельности, основанной на поэтапных действиях и действиях с поддержкой выбора и параллелизма.

На диаграмме деятельности представлены переходы потока управления от одной деятельности к другой внутри системы. Этот вид диаграмм обычно используется для описания поведения, включающего в себя множество параллельных процессов.

Название диаграммы: процесс оформления заказа на сайте

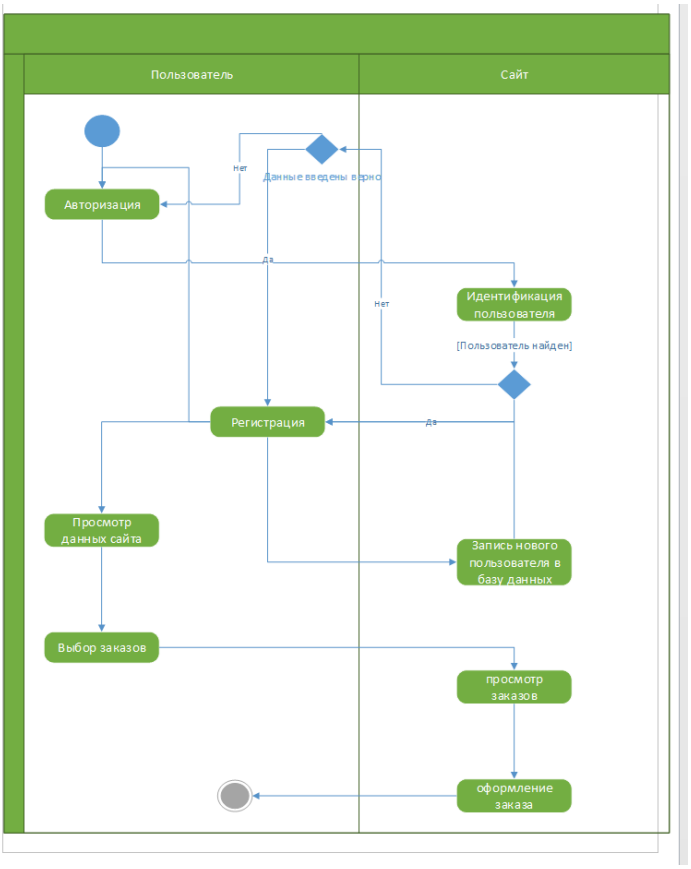

Рисунок 1.3. Диаграмма деятельности процесса оформления заказа на сайте

Описание и анализ диаграммы деятельности

На рисунке 1.3. рассмотрен процесс посещения и оформления заказа на сайте.

Сперва клиент заходит на сайт, смотрит какая услуга ему нужна, далее переходит к ее оформлению.

Администратор в свою очередь может вносить данные в базу данных сайта.

Вывод

В этой главе рассмотрен объектно-ориентированный анализ при помощи UML-диаграмм. Спроектированы 3 вида UML-диаграмм: диаграмма классов, диаграмма последовательностей, диаграмма деятельности. Рассмотрены такие примеры, как магазин по оформлению услуг, процесс заказа и оплаты. Рассмотрев UML-диаграммы на примере, можно сделать вывод, что при помощи унифицированного языка моделирования легче проводить описание процессов и объектно-ориентированный анализ.

2. Объектно-ориентированное программирование

«ООП» — значит «Объектно-Ориентированное Программирование». Это такой подход к написанию программ, который основывается на объектах, а не на функциях и процедурах. Эта модель ставит в центр внимания объекты, а не действия, данные, а не логику. Объект — реализация класса. Все реализации одного класса похожи друг на друга, но могут иметь разные параметры и значения. Объекты могут задействовать методы, специфичные для них. ООП сильно упрощает процесс организации и создания структуры программы. Отдельные объекты, которые можно менять без воздействия на остальные части программы, упрощают также и внесение в программу изменений. Так как с течением времени программы становятся всё более крупными, а их поддержка всё более тяжёлой, эти два аспекта ООП становятся всё более актуальными.

Объектно-Ориентированное Программирование – это одна из парадигм разработки, подразумевающая организацию программного кода, ориентируясь на данные и объекты, а не на функции и логические структуры. Обычно объекты в подобном коде представляют собой полноценные блоки с данными, которые имеют определенный набор характеристик и возможностей.

Условие задачи: Заменить в третьем столбце матрицы A (5×7) все нули на единицы, а в пятом столбце матрицы B (4×5) — все единицы на нули

2.1. Разработка консольного приложения с использованием подпрограмм

2.1.1. Описание функций

Функция **generate_matrix** заполняет матрицу случайными числами от 0 до 9.

Функция **__init__** создает матрицу с заданным числом строк и столбцов

Функция **display** выводит матрицу на экран

Функция **replace_zeros_in_third_column** заменяет все нули в третьем столбце на единицы

Функция **replace_ones_in_fifth_column** заменяет все единицы в пятом столбце на нули

2.1.2. Состав данных консольного приложения с использованием подпрограмм

Таблица 2.1. Состав данных функции **__init__.**

имя	смысл	тип	структура
Исходные данные			
matrix_A	Заданная матрица	целочисленный	Двумерный массив с числом строк 5 и столбцов 7
matrix_B			Двумерный массив с числом строк 4 и столбцов 5
rows	Объявление кол-ва строк	целочисленный	простая переменная
cols	Объявление кол-ва столбцов	целочисленный	простая переменная

Таблица 2.2. Состав данных функции **generate_matrix.**

Имя	**Смысл**	**Тип**	**Структура**
Исходные данные			
matrix_A	Заданная матрица	целый	Двумерный массив с числом строк 5 и столбцов 7
matrix_B	Заданная матрица		Двумерный массив с числом строк 4 и столбцов 5
self.rows	Количество строк	целый	переменная
self.cols	Количество столбцов	целый	переменная
row	Пустой временный список	целый	Пустой список
matrix	Пустой список	целый	Пустой список

Выходные данные			
matrix	Заполненная матрица	целый	Двумерный массив

2.1.3. Шаблон функций и форма ввода-вывода

def __init__(self, rows, cols);

def generate_matrix(self);

def display(self);

def replace_zeros_in_third_column(self);

def replace_ones_in_fifth_column(self);

Input matrix <matrix> with range <row>*<cols>
<matrix[0][0]> < matrix [0][1]> …< matrix [0][cols -1]>
< matrix [1][0]> < matrix [1][1]> …< matrix [1][cols -1]>
…
< matrix [row -1][0]> < matrix [row -1][1]> …< matrix [row -1][cols -1]>

2.1.4. Блок-схема алгоритма функций

Рисунок 2.1. Блок-схема функции main.

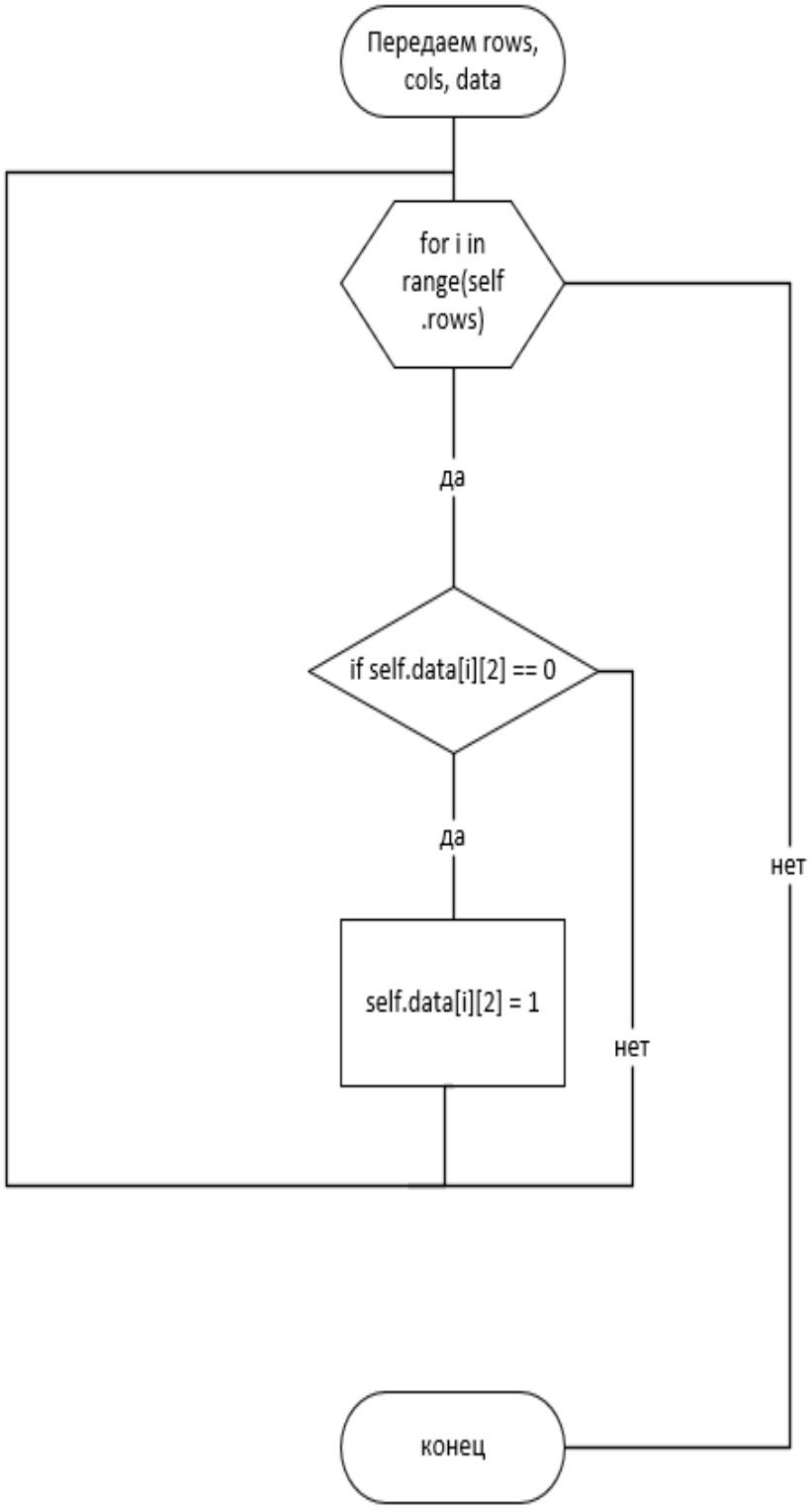

Рисунок 3.2. Блок-схема функции replace_zeros_in_third_column.

2.1.5. Программный код консольного приложения с использованием подпрограмм.

```
'''
Работу выполнил студент группы Иэозс-62-22 Муратов Артемий Денисович
ЗАДАЧА:
Заменить в третьем столбце матрицы A (5×7) все нули на единицы,
 а в пятом столбце матрицы  B (4×5) — все единицы на нули
'''
import random
class Matrix:
    def __init__(self, rows, cols):
        # Создаем матрицу с заданным числом строк и столбцов
        self.rows = rows
        self.cols = cols
        self.data = self.generate_matrix()
    def generate_matrix(self):
        # Заполняем матрицу случайными числами от 0 до 9
        matrix = []
        for i in range(self.rows):
            row = []
            for j in range(self.cols):
                row.append(random.randint(0, 9))
            matrix.append(row)
        return matrix
    def display(self):
        # Выводим матрицу на экран
        for row in self.data:
            print(row)
class ModifiedMatrix(Matrix):
    def replace_zeros_in_third_column(self):
        # Заменяем все нули в третьем столбце на единицы
        for i in range(self.rows):
```

```
            if self.data[i][2] == 0:
                self.data[i][2] = 1
class AnotherModifiedMatrix(Matrix):
    def replace_ones_in_fifth_column(self):
        # Заменяем все единицы в пятом столбце на нули
        for i in range(self.rows):
            if self.data[i][4] == 1:
                self.data[i][4] = 0
if __name__ == "__main__":
    # Создаем матрицу A (5x7) и выводим её до модификации
    matrix_A = ModifiedMatrix(5, 7)
    print("Матрица A до модификации:")
    matrix_A.display()
    # Модифицируем матрицу A
    matrix_A.replace_zeros_in_third_column()
    # Выводим матрицу A после модификации
    print("\nМатрица A после замены нулей в третьем столбце на единицы:")
    matrix_A.display()
    # Создаем матрицу B (4x5) и выводим её до модификации
    matrix_B = AnotherModifiedMatrix(4, 5)
    print("\nМатрица B до модификации:")
    matrix_B.display()
    # Модифицируем матрицу B
    matrix_B.replace_ones_in_fifth_column()
    # Выводим матрицу B после замены единиц в пятом столбце на нули
    print("\nМатрица B после замены единиц в пятом столбце на нули:")
    matrix_B.display()
```

2.1.6. Тестирование программы консольного приложения с использованием подпрограмм.

Ниже показано, что мы заменяем все 0 на 1 в третьем столбце матрицы A (5×7) и все 1 на 0 в пятом столбце матрицы B (4×5)

```
Матрица A до модификации:
[0, 6, 3, 6, 0, 9, 8]
[6, 6, 7, 8, 6, 5, 1]
[6, 2, 7, 8, 9, 8, 9]
[0, 3, 2, 6, 0, 4, 7]
[4, 0, 0, 3, 8, 0, 5]

Матрица A после замены нулей в третьем столбце на единицы:
[0, 6, 3, 6, 0, 9, 8]
[6, 6, 7, 8, 6, 5, 1]
[6, 2, 7, 8, 9, 8, 9]
[0, 3, 2, 6, 0, 4, 7]
[4, 0, 1, 3, 8, 0, 5]

Матрица B до модификации:
[0, 3, 8, 3, 2]
[6, 0, 2, 6, 9]
[7, 5, 1, 5, 2]
[8, 5, 9, 4, 7]

Матрица B после замены единиц в пятом столбце на нули:
[0, 3, 8, 3, 2]
[6, 0, 2, 6, 9]
[7, 5, 1, 5, 2]
[8, 5, 9, 4, 7]
```

Рисунок 4.4. Скриншот результата программы консольного приложения с использованием подпрограмм

2.2. Разработка консольного приложения с использованием классов

2.2.1. Описание класса

Для создания консольного приложения с использованием Объектно-ориентированного подхода был разработан класс Matrix, представляющий абстрактный тип данных «матрица».

В классе реализованы:

- конструктор класса Matrix;
- функция генерации матрицы;
- функция вывода матрицы на экран.

Для реализации принципа наследования был разработаны классы ModifiedMatrix и AnotherModifiedMatrix, которые наследуют класс Matrix.

Для доступа методов класса maxMatrix к полям класса Matrix ко всем полям был применен модификатор доступа protected, обеспечивающий доступ к полям класса всем «наследникам».

2.2.2. Состав данных консольного приложения с использованием классов

Таблица 2.3. Состав данных класса **Matrix**

Имя	**Смысл**	**Тип**
Исходные данные		
rows	количество строк	целочисленные
self.rows		
cols	количество столбцов	
self.cols		
self.data	матрица	
Выходные данные		
matrix	Матрица заполненная случайными числами	целочисленный
row	Временный список для заполнения матрицы	

Таблица 2.4. Состав данных класса **ModifiedMatrix.**

Имя	**Смысл**	**Тип**	**Структура**
Входные данные			
self.data	Матрица, которую меняем	целочисленный	двумерный массив с числом строк 5 и столбцов 7

rows	количество строк	целочисленный	Простая переменная

Также в состав данных класса ModifiedMatrix и AnotherModifiedMatrix входят все поля класса Matrix, так как класс ModifiedMatrix и AnotherModifiedMatrix наследуется от базового класса Matrix.

Таблица 2.5. Состав данных функции **main**

Имя	Смысл	Тип	Структура
Исходные данные			
matrix_A	Экземпляр класса ModifiedMatrix	составной	Экземпляр класса
matrix_B	Экземпляр класса AnotherModifiedMatrix	составной	Экземпляр класса

2.2.3. Блок-схема консольного приложения с использованием классов.

Рисунок 2.3. Блок-схема консольного приложения с использование классов main

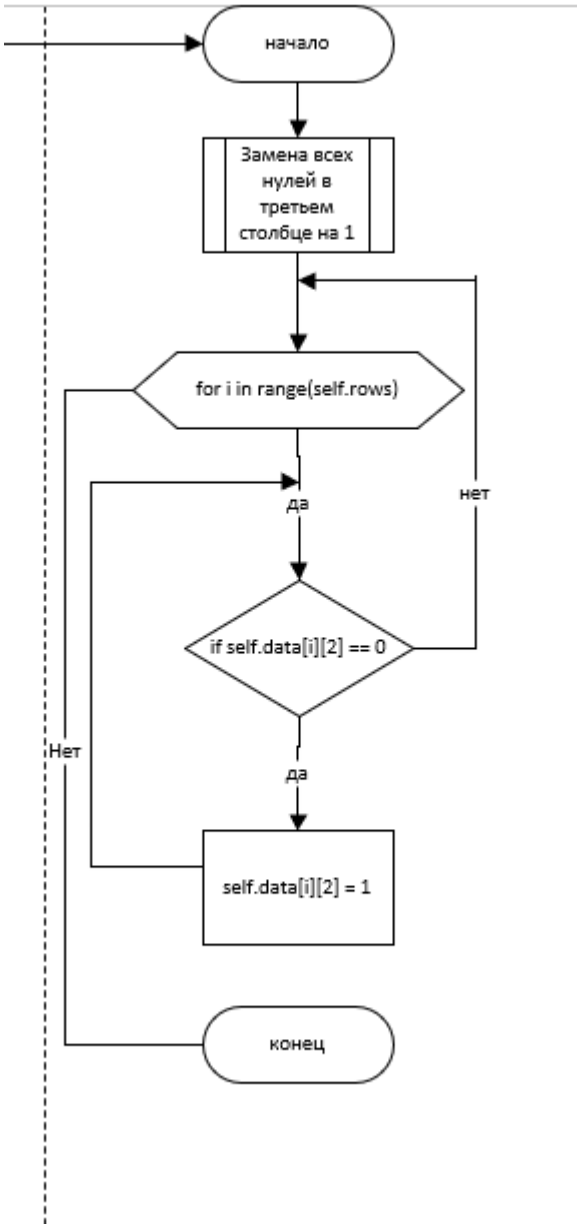

Рисунок 2.4. Блок-схема консольного приложения с использование классов ModifiedMatrix

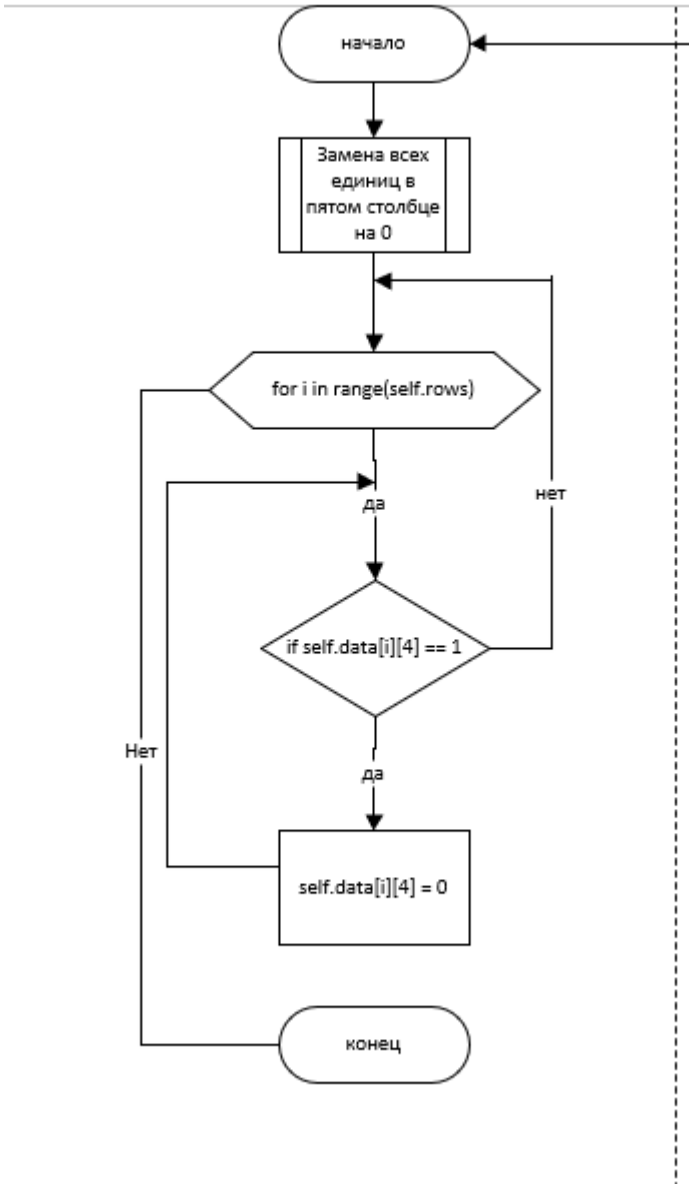

Рисунок 2.5. Блок-схема консольного приложения с использование классов AnotherModifiedMatrix

2.2.4. Программный код консольного приложения с использованием классов.

Код консольного приложения с использованием принципа наследования на языке Python, разработанный в среде VS Code.

```
'''
Работу выполнил студент группы Иэозс-62-22 Муратов Артемий Денисович
```

```
ЗАДАЧА:


Заменить в третьем столбце матрицы A (5×7) все нули на единицы,
а в пятом столбце матрицы  B (4×5) — все единицы на нули
'''
import random
class Matrix:
    def __init__(self, rows, cols):
        # Создаем матрицу с заданным числом строк и столбцов
        self.rows = rows
        self.cols = cols
        self.data = self.generate_matrix()
    def generate_matrix(self):
        # Заполняем матрицу случайными числами от 0 до 9
        matrix = []
        for i in range(self.rows):
            row = []
            for j in range(self.cols):
                row.append(random.randint(0, 9))
            matrix.append(row)
        return matrix
    def display(self):
        # Выводим матрицу на экран
        for row in self.data:
            print(row)
class ModifiedMatrix(Matrix):
    def replace_zeros_in_third_column(self):
        # Заменяем все нули в третьем столбце на единицы
        for i in range(self.rows):
            if self.data[i][2] == 0:
                self.data[i][2] = 1
class AnotherModifiedMatrix(Matrix):
```

```
    def replace_ones_in_fifth_column(self):
        # Заменяем все единицы в пятом столбце на нули
        for i in range(self.rows):
            if self.data[i][4] == 1:
                self.data[i][4] = 0
if __name__ == "__main__":
    # Создаем матрицу A (5x7) и выводим её до модификации
    matrix_A = ModifiedMatrix(5, 7)
    print("Матрица A до модификации:")
    matrix_A.display()
    # Модифицируем матрицу A
    matrix_A.replace_zeros_in_third_column()
    # Выводим матрицу A после модификации
    print("\nМатрица A после замены нулей в третьем столбце на единицы:")
    matrix_A.display()
    # Создаем матрицу B (4x5) и выводим её до модификации
    matrix_B = AnotherModifiedMatrix(4, 5)
    print("\nМатрица B до модификации:")
    matrix_B.display()
    # Модифицируем матрицу B
    matrix_B.replace_ones_in_fifth_column()
    # Выводим матрицу B после замены единиц в пятом столбце на нули
    print("\nМатрица B после замены единиц в пятом столбце на нули:")
    matrix_B.display()
```

2.2.7. Тестирование программы консольного приложения с использованием подпрограмм.

На рисунке 2.6 представлен скриншот проводимого тестирования программы консольного приложения с использованием классов. Ниже мы можем видеть, что мы заменяем все 0 на 1 в третьем столбце матрицы A (5×7) и в пятом столбце матрицы B (4×5) — все единицы на нули

```
Матрица A до модификации:
[8, 1, 9, 0, 9, 1, 1]
[1, 6, 2, 4, 4, 3, 2]
[6, 4, 6, 4, 9, 7, 4]
[1, 2, 3, 6, 9, 4, 1]
[2, 0, 0, 1, 1, 5, 0]

Матрица A после замены нулей в третьем столбце на единицы:
[8, 1, 9, 0, 9, 1, 1]
[1, 6, 2, 4, 4, 3, 2]
[6, 4, 6, 4, 9, 7, 4]
[1, 2, 3, 6, 9, 4, 1]
[2, 0, 1, 1, 1, 5, 0]

Матрица B до модификации:
[2, 8, 0, 4, 7]
[7, 1, 6, 6, 0]
[3, 2, 1, 8, 1]
[7, 8, 4, 7, 5]

Матрица B после замены единиц в пятом столбце на нули:
[2, 8, 0, 4, 7]
[7, 1, 6, 6, 0]
[3, 2, 1, 8, 0]
[7, 8, 4, 7, 5]
```

Рисунок 2.6. Скриншот результата программы со всем заданными матрицами

2.3. Разработка оконного приложения

2.3.1. Процесс разработки оконного приложения

Для разработки оконного приложения (рис. 2.7) использовался язык Python и Framework PySide6 с использованием интерфейса QT designer

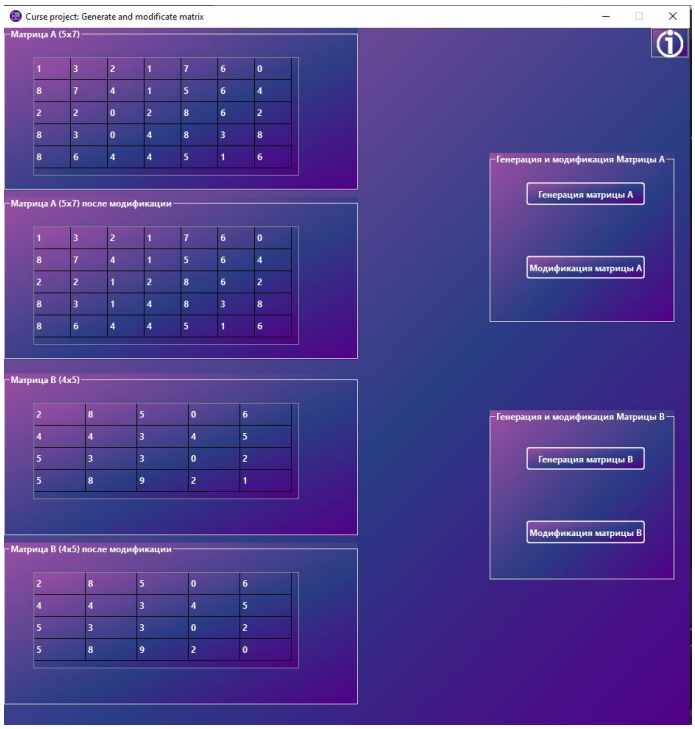

Рисунок 2.7. Оконная форма работы программы

Для ввода двух матриц: A (5×7), B (4×5) используется компонент Button (1, 3), который генерирует случайные значения для каждой матрицы с

диапазоном, заданным по условию задачи и выводит результат в QTableWidget (1, 3).

Далее при нажатии на кнопки Button (2, 4), программа изменяет матрицы, в соответствии с условием задачи, и выводит результат в QTableWidget (2, 4) на форму. Если в сгенерированной матрице отсутствует 1 или 0, то на экран выводится MessageBox с текстом: «В сгенерированной матрице А в 3м столбце нет 0 Пожалуйста, создайте новую матрицу А!» - смотреть рисунок 2.8 или «В сгенерированной матрице В в 5м столбце нет 1 Пожалуйста, создайте новую матрицу В!» - смотреть рисунок 2.9

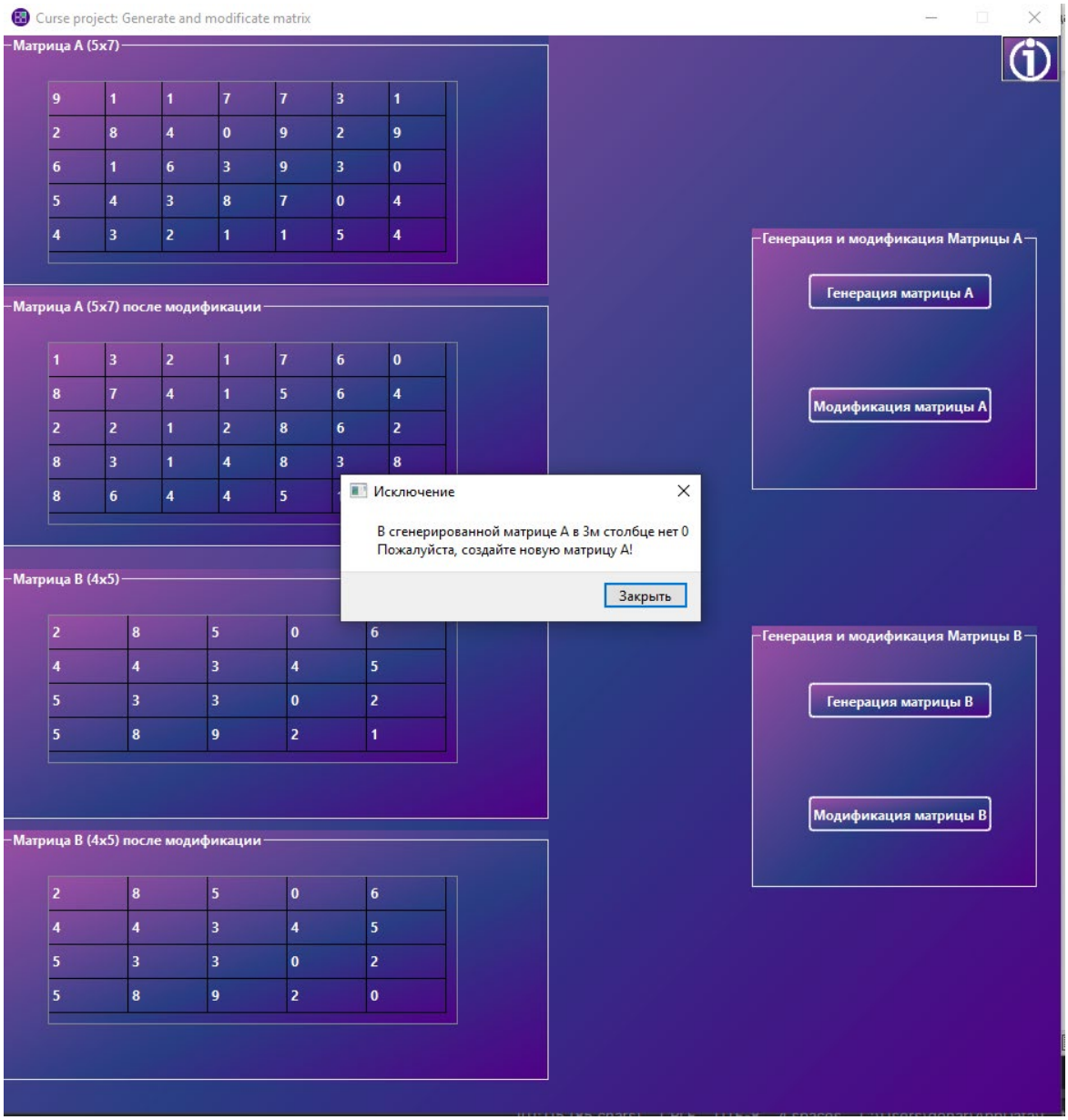

Рисунок 2.8. Примет вывода сообщения на экран для матрицы А

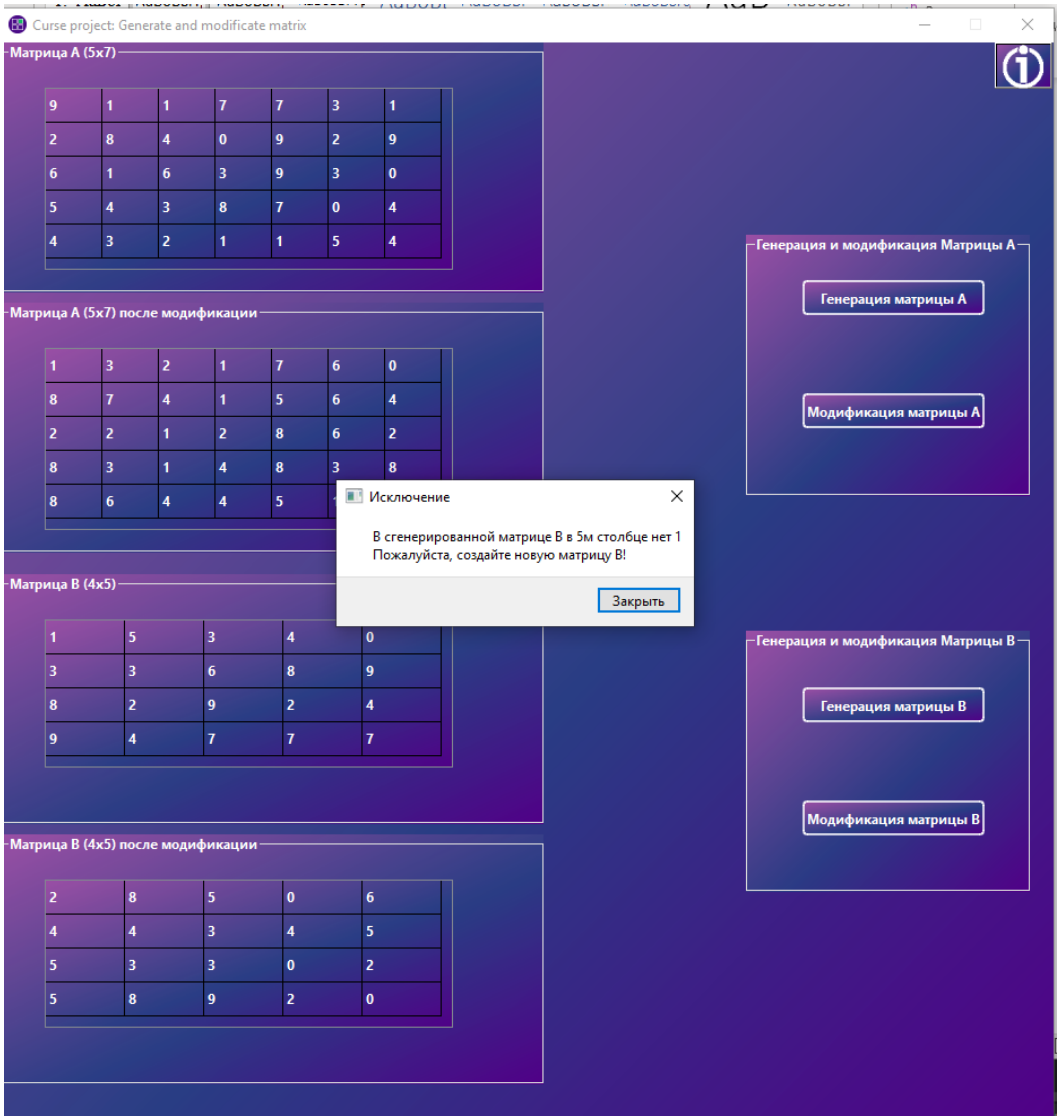

Рисунок 2.9. Примет вывода сообщения на экран для матрицы B

Для курсового проекта мною были созданы favicon и иконка информации, в программе Adobe Photoshop, посмотреть в близи можно на рисунках 2.10 и 2.11

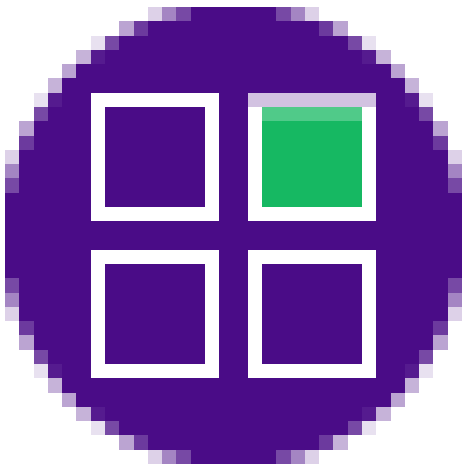

Рисунок 2.10. Favicon

Рисунок 2.11. Иконка info

Программный код файла class_matrix

В данном файле прописаны класс генерации матрицы и 2 класса изменения матрицы, по условию задачи.

```
'''
Работу выполнил студент группы Иэозс-62-22 Муратов Артемий Денисович

ЗАДАЧА:

Заменить в третьем столбце матрицы A (5×7) все нули на единицы,
а в пятом столбце матрицы  B (4×5) — все единицы на нули
'''
import random

class Matrix:
    def __init__(self, rows, cols):
        # Создаем матрицу с заданным числом строк и столбцов
        self.rows = rows
        self.cols = cols
        self.data = self.generate_matrix()

    def generate_matrix(self):
        # Заполняем матрицу случайными числами от 0 до 9
        matrix = []
        for i in range(self.rows):
            row = []
            for j in range(self.cols):
                row.append(random.randint(0, 9))
            matrix.append(row)
        return matrix

    def display(self):
        # Выводим матрицу на экран
        for row in self.data:
            print(row)

class ModifiedMatrix(Matrix):
    def replace_zeros_in_third_column(self):
```

```
        # Заменяем все нули в третьем столбце на единицы
        for i in range(self.rows):
            if self.data[i][2] == 0:
                self.data[i][2] = 1

class AnotherModifiedMatrix(Matrix):
    def replace_ones_in_fifth_column(self):
        # Заменяем все единицы в пятом столбце на нули
        for i in range(self.rows):
            if self.data[i][4] == 1:
                self.data[i][4] = 0

"""
if __name__ == "__main__":
    # Создаем матрицу A (5x7) и выводим её до модификации
    matrix_A = ModifiedMatrix(5, 7)
    print("Матрица A до модификации:")
    matrix_A.display()

    # Модифицируем матрицу A
    matrix_A.replace_zeros_in_third_column()

    # Выводим матрицу A после модификации
    print("\nМатрица A после замены нулей в третьем столбце на единицы:")
    matrix_A.display()

    # Создаем матрицу B (4x5) и выводим её до модификации
    matrix_B = AnotherModifiedMatrix(4, 5)
    print("\nМатрица B до модификации:")
    matrix_B.display()

    # Модифицируем матрицу B
    matrix_B.replace_ones_in_fifth_column()

    # Выводим матрицу B после замены единиц в пятом столбце на нули
    print("\nМатрица B после замены единиц в пятом столбце на нули:")
    matrix_B.display()
"""
```

Программный код файла graphical_user_interface

В данном файле прописан весь дизайн оконного приложения, размещение всех элементов и их графические свойства.

```
from PySide6.QtCore import (QCoreApplication, QMetaObject, QRect,
                            QSize)
from PySide6.QtGui import (QIcon)
from PySide6.QtWidgets import (QGroupBox, QPushButton, QTableWidget, QTableWidgetItem,
                               QWidget)


class Ui_MainWindow(object):
```

```
    def __init__(self):
        self.btn_info = None
        self.tableWidget_matrixB = None
        self.groupBox_5 = None
        self.tableWidget_new_matrixB = None
        self.groupBox_6 = None
        self.tableWidget_new_matrixA = None
        self.groupBox_4 = None
        self.tableWidget_matrixA = None
        self.groupBox_3 = None
        self.btn_modificate_matrixB = None
        self.btn_generate_matrixB = None
        self.groupBox_2 = None
        self.btn_modificate_matrixA = None
        self.btn_generate_matrixA = None
        self.groupBox = None
        self.centralwidget = None

    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(932, 949)
        MainWindow.setMinimumSize(QSize(932, 949))
        MainWindow.setMaximumSize(QSize(932, 949))
        icon = QIcon()
        icon.addFile(u"img/favicon.gif", QSize(), QIcon.Normal, QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setStyleSheet(u"background-color: qlineargradient(spread:pad, x1:1, y1:1, x2:0, y2:0,
stop:0 rgba("
                                 u"81, 0, 135, 255), stop:0.427447 rgba(41, 61, 132, 235), stop:1 rgba(155, 79, 165, "
                                 u"255));")
        MainWindow.setIconSize(QSize(24, 24))
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
        self.groupBox = QGroupBox(self.centralwidget)
        self.groupBox.setObjectName(u"groupBox")
        self.groupBox.setGeometry(QRect(660, 170, 251, 231))
        self.groupBox.setStyleSheet(u"color: white;\n"
                                    "font-weight: bold;\n"
                                    "font-size:15;\n"
                                    "")
        self.btn_generate_matrixA = QPushButton(self.groupBox)
        self.btn_generate_matrixA.setObjectName(u"btn_generate_matrixA")
        self.btn_generate_matrixA.setGeometry(QRect(50, 40, 161, 31))
        self.btn_generate_matrixA.setStyleSheet(u"QPushButton{\n"
                                                "border: 2px solid rgba(255, 255, 255, 200);\n"
                                                "border-radius: 4px;\n"
                                                "}\n"
                                                "QPushButton:pressed\n"
                                                "{\n"
                                                "background-color: rgba(22, 184, 98,1)\n"
                                                "}")
        self.btn_modificate_matrixA = QPushButton(self.groupBox)
        self.btn_modificate_matrixA.setObjectName(u"btn_modificate_matrixA")
        self.btn_modificate_matrixA.setGeometry(QRect(50, 140, 161, 31))
        self.btn_modificate_matrixA.setStyleSheet(u"QPushButton{\n"
                                                  "border: 2px solid rgba(255, 255, 255, 200);\n"
```

```
                                "border-radius: 4px;\n"
                                "}\n"
                                "QPushButton:pressed\n"
                                "{\n"
                                "background-color: rgba(22, 184, 98,1)\n"
                                "}")
        self.groupBox_2 = QGroupBox(self.centralwidget)
        self.groupBox_2.setObjectName(u"groupBox_2")
        self.groupBox_2.setGeometry(QRect(660, 520, 251, 231))
        self.groupBox_2.setStyleSheet(u"color: white;\n"
                            "font-weight: bold;\n"
                            "font-size:15;\n"
                            "")
        self.btn_generate_matrixB = QPushButton(self.groupBox_2)
        self.btn_generate_matrixB.setObjectName(u"btn_generate_matrixB")
        self.btn_generate_matrixB.setGeometry(QRect(50, 50, 161, 31))
        self.btn_generate_matrixB.setStyleSheet(u"QPushButton{\n"
                                "border: 2px solid rgba(255, 255, 255, 200);\n"
                                "border-radius: 4px;\n"
                                "}\n"
                                "QPushButton:pressed\n"
                                "{\n"
                                "background-color: rgba(22, 184, 98,1)\n"
                                "}")
        self.btn_modificate_matrixB = QPushButton(self.groupBox_2)
        self.btn_modificate_matrixB.setObjectName(u"btn_modificate_matrixB")
        self.btn_modificate_matrixB.setGeometry(QRect(50, 150, 161, 31))
        self.btn_modificate_matrixB.setStyleSheet(u"QPushButton{\n"
                                "border: 2px solid rgba(255, 255, 255, 200);\n"
                                "border-radius: 4px;\n"
                                "}\n"
                                "QPushButton:pressed\n"
                                "{\n"
                                "background-color: rgba(22, 184, 98,1)\n"
                                "}")
        self.groupBox_3 = QGroupBox(self.centralwidget)
        self.groupBox_3.setObjectName(u"groupBox_3")
        self.groupBox_3.setGeometry(QRect(0, 0, 481, 221))
        self.groupBox_3.setStyleSheet(u"color: white;\n"
                            "font-weight: bold;\n"
                            "font-size:15;\n"
                            "")
        self.tableWidget_matrixA = QTableWidget(self.groupBox_3)
        if self.tableWidget_matrixA.columnCount() < 7:
            self.tableWidget_matrixA.setColumnCount(7)
        __qtablewidgetitem = QTableWidgetItem()
        self.tableWidget_matrixA.setHorizontalHeaderItem(0, __qtablewidgetitem)
        if self.tableWidget_matrixA.rowCount() < 5:
            self.tableWidget_matrixA.setRowCount(5)
        __qtablewidgetitem1 = QTableWidgetItem()
        self.tableWidget_matrixA.setVerticalHeaderItem(0, __qtablewidgetitem1)
        __qtablewidgetitem2 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 0, __qtablewidgetitem2)
        __qtablewidgetitem3 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 1, __qtablewidgetitem3)
        __qtablewidgetitem4 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 2, __qtablewidgetitem4)
```

```
        __qtablewidgetitem5 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 3, __qtablewidgetitem5)
        __qtablewidgetitem6 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 4, __qtablewidgetitem6)
        __qtablewidgetitem7 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 5, __qtablewidgetitem7)
        __qtablewidgetitem8 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(0, 6, __qtablewidgetitem8)
        __qtablewidgetitem9 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 0, __qtablewidgetitem9)
        __qtablewidgetitem10 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 1, __qtablewidgetitem10)
        __qtablewidgetitem11 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 2, __qtablewidgetitem11)
        __qtablewidgetitem12 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 3, __qtablewidgetitem12)
        __qtablewidgetitem13 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 4, __qtablewidgetitem13)
        __qtablewidgetitem14 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 5, __qtablewidgetitem14)
        __qtablewidgetitem15 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(1, 6, __qtablewidgetitem15)
        __qtablewidgetitem16 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(2, 0, __qtablewidgetitem16)
        __qtablewidgetitem17 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(2, 1, __qtablewidgetitem17)
        __qtablewidgetitem18 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(3, 0, __qtablewidgetitem18)
        __qtablewidgetitem19 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(3, 1, __qtablewidgetitem19)
        __qtablewidgetitem20 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(4, 0, __qtablewidgetitem20)
        __qtablewidgetitem21 = QTableWidgetItem()
        self.tableWidget_matrixA.setItem(4, 1, __qtablewidgetitem21)
        self.tableWidget_matrixA.setObjectName(u"tableWidget_matrixA")
        self.tableWidget_matrixA.setGeometry(QRect(40, 40, 361, 161))
        self.tableWidget_matrixA.setStyleSheet(u"color: white;\n"
                                    "")
        self.tableWidget_matrixA.setShowGrid(True)
        self.tableWidget_matrixA.setRowCount(5)
        self.tableWidget_matrixA.setColumnCount(7)
        self.tableWidget_matrixA.horizontalHeader().setVisible(False)
        self.tableWidget_matrixA.horizontalHeader().setMinimumSectionSize(50)
        self.tableWidget_matrixA.horizontalHeader().setDefaultSectionSize(50)
        self.tableWidget_matrixA.verticalHeader().setVisible(False)
        self.groupBox_4 = QGroupBox(self.centralwidget)
        self.groupBox_4.setObjectName(u"groupBox_4")
        self.groupBox_4.setGeometry(QRect(0, 230, 481, 221))
        self.groupBox_4.setStyleSheet(u"color: white;\n"
                            "font-weight: bold;\n"
                            "font-size:15;\n"
                            "")
        self.tableWidget_new_matrixA = QTableWidget(self.groupBox_4)
        if self.tableWidget_new_matrixA.columnCount() < 7:
            self.tableWidget_new_matrixA.setColumnCount(7)
        if self.tableWidget_new_matrixA.rowCount() < 5:
            self.tableWidget_new_matrixA.setRowCount(5)
```

```
        __qtablewidgetitem22 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 0, __qtablewidgetitem22)
        __qtablewidgetitem23 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 1, __qtablewidgetitem23)
        __qtablewidgetitem24 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 2, __qtablewidgetitem24)
        __qtablewidgetitem25 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 3, __qtablewidgetitem25)
        __qtablewidgetitem26 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 4, __qtablewidgetitem26)
        __qtablewidgetitem27 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 5, __qtablewidgetitem27)
        __qtablewidgetitem28 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(0, 6, __qtablewidgetitem28)
        __qtablewidgetitem29 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(1, 0, __qtablewidgetitem29)
        __qtablewidgetitem30 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(2, 0, __qtablewidgetitem30)
        __qtablewidgetitem31 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(3, 0, __qtablewidgetitem31)
        __qtablewidgetitem32 = QTableWidgetItem()
        self.tableWidget_new_matrixA.setItem(4, 0, __qtablewidgetitem32)
        self.tableWidget_new_matrixA.setObjectName(u"tableWidget_new_matrixA")
        self.tableWidget_new_matrixA.setGeometry(QRect(40, 40, 361, 161))
        self.tableWidget_new_matrixA.setStyleSheet(u"color: white;\n"
                                    "")
        self.tableWidget_new_matrixA.setRowCount(5)
        self.tableWidget_new_matrixA.setColumnCount(7)
        self.tableWidget_new_matrixA.horizontalHeader().setVisible(False)
        self.tableWidget_new_matrixA.horizontalHeader().setDefaultSectionSize(50)
        self.tableWidget_new_matrixA.horizontalHeader().setStretchLastSection(False)
        self.tableWidget_new_matrixA.verticalHeader().setVisible(False)
        self.tableWidget_new_matrixA.verticalHeader().setHighlightSections(True)
        self.tableWidget_new_matrixA.verticalHeader().setStretchLastSection(False)
        self.groupBox_6 = QGroupBox(self.centralwidget)
        self.groupBox_6.setObjectName(u"groupBox_6")
        self.groupBox_6.setGeometry(QRect(0, 700, 481, 221))
        self.groupBox_6.setStyleSheet(u"color: white;\n"
                            "font-weight: bold;\n"
                            "font-size:15;\n"
                            "")
        self.tableWidget_new_matrixB = QTableWidget(self.groupBox_6)
        if self.tableWidget_new_matrixB.columnCount() < 5:
            self.tableWidget_new_matrixB.setColumnCount(5)
        if self.tableWidget_new_matrixB.rowCount() < 4:
            self.tableWidget_new_matrixB.setRowCount(4)
        __qtablewidgetitem33 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(0, 0, __qtablewidgetitem33)
        __qtablewidgetitem34 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(0, 1, __qtablewidgetitem34)
        __qtablewidgetitem35 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(0, 2, __qtablewidgetitem35)
        __qtablewidgetitem36 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(0, 3, __qtablewidgetitem36)
        __qtablewidgetitem37 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(0, 4, __qtablewidgetitem37)
        __qtablewidgetitem38 = QTableWidgetItem()
```

```
        self.tableWidget_new_matrixB.setItem(1, 0, __qtablewidgetitem38)
        __qtablewidgetitem39 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(2, 0, __qtablewidgetitem39)
        __qtablewidgetitem40 = QTableWidgetItem()
        self.tableWidget_new_matrixB.setItem(3, 0, __qtablewidgetitem40)
        self.tableWidget_new_matrixB.setObjectName(u"tableWidget_new_matrixB")
        self.tableWidget_new_matrixB.setGeometry(QRect(40, 40, 361, 131))
        self.tableWidget_new_matrixB.setStyleSheet(u"color: white;\n"
                                    "")
        self.tableWidget_new_matrixB.setRowCount(4)
        self.tableWidget_new_matrixB.setColumnCount(5)
        self.tableWidget_new_matrixB.horizontalHeader().setVisible(False)
        self.tableWidget_new_matrixB.horizontalHeader().setCascadingSectionResizes(False)
        self.tableWidget_new_matrixB.horizontalHeader().setMinimumSectionSize(70)
        self.tableWidget_new_matrixB.horizontalHeader().setDefaultSectionSize(70)
        self.tableWidget_new_matrixB.horizontalHeader().setProperty("showSortIndicator", False)
        self.tableWidget_new_matrixB.horizontalHeader().setStretchLastSection(False)
        self.tableWidget_new_matrixB.verticalHeader().setVisible(False)
        self.tableWidget_new_matrixB.verticalHeader().setStretchLastSection(False)
        self.groupBox_5 = QGroupBox(self.centralwidget)
        self.groupBox_5.setObjectName(u"groupBox_5")
        self.groupBox_5.setGeometry(QRect(0, 470, 481, 221))
        self.groupBox_5.setStyleSheet(u"color: white;\n"
                            "font-weight: bold;\n"
                            "font-size:15;\n"
                            "")
        self.tableWidget_matrixB = QTableWidget(self.groupBox_5)
        if self.tableWidget_matrixB.columnCount() < 5:
            self.tableWidget_matrixB.setColumnCount(5)
        if self.tableWidget_matrixB.rowCount() < 4:
            self.tableWidget_matrixB.setRowCount(4)
        __qtablewidgetitem41 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(0, 0, __qtablewidgetitem41)
        __qtablewidgetitem42 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(0, 1, __qtablewidgetitem42)
        __qtablewidgetitem43 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(0, 2, __qtablewidgetitem43)
        __qtablewidgetitem44 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(0, 3, __qtablewidgetitem44)
        __qtablewidgetitem45 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(0, 4, __qtablewidgetitem45)
        __qtablewidgetitem46 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(1, 0, __qtablewidgetitem46)
        __qtablewidgetitem47 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(2, 0, __qtablewidgetitem47)
        __qtablewidgetitem48 = QTableWidgetItem()
        self.tableWidget_matrixB.setItem(3, 0, __qtablewidgetitem48)
        self.tableWidget_matrixB.setObjectName(u"tableWidget_matrixB")
        self.tableWidget_matrixB.setGeometry(QRect(40, 40, 361, 131))
        self.tableWidget_matrixB.setStyleSheet(u"color: white;\n"
                                    "")
        self.tableWidget_matrixB.setRowCount(4)
        self.tableWidget_matrixB.setColumnCount(5)
        self.tableWidget_matrixB.horizontalHeader().setVisible(False)
        self.tableWidget_matrixB.horizontalHeader().setDefaultSectionSize(70)
        self.tableWidget_matrixB.horizontalHeader().setStretchLastSection(False)
        self.tableWidget_matrixB.verticalHeader().setVisible(False)
```

```
        self.tableWidget_matrixB.verticalHeader().setStretchLastSection(False)
        self.btn_info = QPushButton(self.centralwidget)
        self.btn_info.setObjectName(u"btn_info")
        self.btn_info.setEnabled(True)
        self.btn_info.setGeometry(QRect(880, 0, 51, 41))
        icon1 = QIcon()
        icon1.addFile(u"img/matrinfo.png", QSize(), QIcon.Normal, QIcon.Off)
        self.btn_info.setIcon(icon1)
        self.btn_info.setIconSize(QSize(40, 40))
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)

        QMetaObject.connectSlotsByName(MainWindow)

    # setupUi

    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(
            QCoreApplication.translate("MainWindow", u"Curse project: Generate and modificate matrix",
None))
        self.groupBox.setTitle(QCoreApplication.translate("MainWindow",
                                          u"\u0413\u0435\u043d\u0435\u0440\u0430\u0446\u0438\u044f "
                                          u"\u0438 "
                                          u"\u043c\u043e\u0434\u0438\u0444\u0438\u043a\u0430\u0446"
                                          u"\u0438\u044f \u041c\u0430\u0442\u0440\u0438\u0446\u044b "
                                          u"\u0410",
                                          None))
        self.btn_generate_matrixA.setText(QCoreApplication.translate("MainWindow",
                                              u"\u0413\u0435\u043d\u0435\u0440\u0430\u0446"
                                              u"\u0438\u044f "
                                              u"\u043c\u0430\u0442\u0440\u0438\u0446\u044b "
                                              u"\u0410",
                                              None))
        self.btn_modificate_matrixA.setText(QCoreApplication.translate("MainWindow",
                                               u"\u041c\u043e\u0434\u0438\u0444\u0438\u043a"
                                               u"\u0430\u0446\u0438\u044f "
                                               u"\u043c\u0430\u0442\u0440\u0438\u0446\u044b "
                                               u"\u0410",
                                               None))
        self.groupBox_2.setTitle(QCoreApplication.translate("MainWindow",
                                            u"\u0413\u0435\u043d\u0435\u0440\u0430\u0446\u0438\u044f "
                                            u"\u0438 "
                                            u"\u043c\u043e\u0434\u0438\u0444\u0438\u043a\u0430\u0446"
                                            u"\u0438\u044f \u041c\u0430\u0442\u0440\u0438\u0446\u044b "
                                            u"B",
                                            None))
        self.btn_generate_matrixB.setText(QCoreApplication.translate("MainWindow",
                                              u"\u0413\u0435\u043d\u0435\u0440\u0430\u0446"
                                              u"\u0438\u044f "
                                              u"\u043c\u0430\u0442\u0440\u0438\u0446\u044b B",
                                              None))
        self.btn_modificate_matrixB.setText(QCoreApplication.translate("MainWindow",
                                               u"\u041c\u043e\u0434\u0438\u0444\u0438\u043a"
                                               u"\u0430\u0446\u0438\u044f "
                                               u"\u043c\u0430\u0442\u0440\u0438\u0446\u044b B",
                                               None))
```

```
        self.groupBox_3.setTitle(
            QCoreApplication.translate("MainWindow", u"\u041c\u0430\u0442\u0440\u0438\u0446\u0430
\u0410 (5x7)", None))

        __sortingEnabled = self.tableWidget_matrixA.isSortingEnabled()
        self.tableWidget_matrixA.setSortingEnabled(False)
        self.tableWidget_matrixA.setSortingEnabled(__sortingEnabled)

        self.groupBox_4.setTitle(QCoreApplication.translate("MainWindow",
                                            u"\u041c\u0430\u0442\u0440\u0438\u0446\u0430 \u0410 (5x7) "
                                            u"\u043f\u043e\u0441\u043b\u0435 "
                                            u"\u043c\u043e\u0434\u0438\u0444\u0438\u043a\u0430\u0446"
                                            u"\u0438\u0438",
                                            None))

        __sortingEnabled1 = self.tableWidget_new_matrixA.isSortingEnabled()
        self.tableWidget_new_matrixA.setSortingEnabled(False)
        self.tableWidget_new_matrixA.setSortingEnabled(__sortingEnabled1)

        self.groupBox_6.setTitle(QCoreApplication.translate("MainWindow",
                                            u"\u041c\u0430\u0442\u0440\u0438\u0446\u0430 \u0412 (4x5) "
                                            u"\u043f\u043e\u0441\u043b\u0435 "
                                            u"\u043c\u043e\u0434\u0438\u0444\u0438\u043a\u0430\u0446"
                                            u"\u0438\u0438",
                                            None))

        __sortingEnabled2 = self.tableWidget_new_matrixB.isSortingEnabled()
        self.tableWidget_new_matrixB.setSortingEnabled(False)
        self.tableWidget_new_matrixB.setSortingEnabled(__sortingEnabled2)

        self.groupBox_5.setTitle(
            QCoreApplication.translate("MainWindow", u"\u041c\u0430\u0442\u0440\u0438\u0446\u0430
\u0412 (4x5)", None))

        __sortingEnabled3 = self.tableWidget_matrixB.isSortingEnabled()
        self.tableWidget_matrixB.setSortingEnabled(False)
        self.tableWidget_matrixB.setSortingEnabled(__sortingEnabled3)

        self.btn_info.setText("")
    # retranslateUi
```

Программный код файла pr7_curse_project

Данный файл является основным, для запуска приложения запускаем его. В этот файл импортируем два вышеописанных. В основном классе прописываем отрисовку окна и всех его компонентов, также прописываем действия для всех кнопок и виджетов.

```
"""
Работу выполнил студент группы Иэозс-62-22 Муратов Артемий Денисович

ЗАДАЧА:
Разработать оконное приложение для консольной программы из пр56.
Условие задачи из пр56:

Заменить в третьем столбце матрицы A (5×7) все нули на единицы,
а в пятом столбце матрицы  B (4×5) — все единицы на нули
"""
import sys

from PySide6 import QtWidgets
from PySide6.QtWidgets import QTableWidgetItem

from class_matrix import ModifiedMatrix, AnotherModifiedMatrix
from grafical_user_interface import Ui_MainWindow


class CurseProjectWinApp(QtWidgets.QMainWindow):
    def __init__(self):
        # super - возвращает объект родителя класса CurseProjectWinApp  и вызывает его конструктор
        super(CurseProjectWinApp, self).__init__()
        # экземпляры для отрисовки окна
        self.matrB = None
        self.matrA = None
        self.gui = Ui_MainWindow()
        self.gui.setupUi(self)
        self.init_GUI()

    # функция заполнения таблицы tableWidget_matrixA сгенерированной матрицей A(5x7)
    def input_martixA_in_tableA(self):
        self.matrA = ModifiedMatrix(5, 7)
        self.gui.tableWidget_matrixA.setRowCount(self.matrA.rows)
        self.gui.tableWidget_matrixA.setColumnCount(self.matrA.cols)
        for i in range(self.matrA.rows):
            for j in range(self.matrA.cols):
                item = QTableWidgetItem(str(self.matrA.data[i][j]))
                self.gui.tableWidget_matrixA.setItem(i, j, item)

    # функция замены всех нулей в третьем столбце на единицы
    def input_matrixA_in_modificate_tableA(self):
        # булевая переменная - флаг, с помощью которого будет проводится проверка условия
        zero_in_third_column = False
        # проверяем элементы 3го столбца,  есть ли 0, если есть то меняем флаг и выходим досрочно
        for i in range(self.matrA.rows):
            if self.matrA.data[i][2] == 0:
                zero_in_third_column = True
                break
        # если флаг True,  то меняем 0 на 1 и заполняем таблицу
        if zero_in_third_column:
            self.matrA.replace_zeros_in_third_column()
            self.gui.tableWidget_new_matrixA.setRowCount(self.matrA.rows)
            self.gui.tableWidget_new_matrixA.setColumnCount(self.matrA.cols)
            for i in range(self.matrA.rows):
                for j in range(self.matrA.cols):
                    item = QTableWidgetItem(str(self.matrA.data[i][j]))
```

```
                self.gui.tableWidget_new_matrixA.setItem(i, j, item)

        # иначе выводим сообщение, что 0 в столбце нет
        else:
            msg_box = QtWidgets.QMessageBox()
            msg_box.setWindowTitle("Исключение")
            msg_box.setText("В сгенерированной матрице A в 3м столбце нет 0\nПожалуйста, создайте
новую матрицу A!")
            close_button = QtWidgets.QPushButton("Закрыть")
            msg_box.addButton(close_button, QtWidgets.QMessageBox.AcceptRole)
            msg_box.exec_()

    # функция заполнения таблицы tableWidget_matrixB сгенерированной матрицей B(4x5)
    def input_martixB_in_tableB(self):
        self.matrB = AnotherModifiedMatrix(4, 5)
        self.gui.tableWidget_matrixB.setRowCount(self.matrB.rows)
        self.gui.tableWidget_matrixB.setColumnCount(self.matrB.cols)
        for i in range(self.matrB.rows):
            for j in range(self.matrB.cols):
                item = QTableWidgetItem(str(self.matrB.data[i][j]))
                self.gui.tableWidget_matrixB.setItem(i, j, item)

    # функция для замены всех единиц в пятом столбце на нули
    def input_matrixB_in_modificate_tableB(self):
        # булевая переменная - флаг, с помощью которого будет проводится проверка условия
        ed_in_five_columns = False
        # проверяем элементы 5го столбца,  есть ли 1, если есть то меняем флаг и выходим досрочно
        for i in range(self.matrB.rows):
            if self.matrB.data[i][4] == 1:
                ed_in_five_columns = True
                break
        # если флаг True,  то меняем 1 на 0 и заполняем таблицу
        if ed_in_five_columns:
            self.matrB.replace_ones_in_fifth_column()
            self.gui.tableWidget_new_matrixB.setRowCount(self.matrB.rows)
            self.gui.tableWidget_new_matrixB.setColumnCount(self.matrB.cols)
            for i in range(self.matrB.rows):
                for j in range(self.matrB.cols):
                    item = QTableWidgetItem(str(self.matrB.data[i][j]))
                    self.gui.tableWidget_new_matrixB.setItem(i, j, item)
        # иначе выводим сообщение, что 1 в столбце нет
        else:
            msg_box = QtWidgets.QMessageBox()
            msg_box.setWindowTitle("Исключение")
            msg_box.setText("В сгенерированной матрице B в 5м столбце нет 1\nПожалуйста, создайте
новую матрицу B!")
            close_button = QtWidgets.QPushButton("Закрыть")
            msg_box.addButton(close_button, QtWidgets.QMessageBox.AcceptRole)
            msg_box.exec_()

    # функция для изменения интерфейса
    def init_GUI(self):

        # Всплывающая подсказка при наведении курсора на кнопку, в которой находится
изображение
        self.gui.btn_info.setToolTip("Работу выполнил студент группы Иэозс-62-22 Муратов Артемий
Денисович.\n"
```

```
                        " ЗАДАЧА:\n"
                        "Разработать оконное приложение для консольной программы из пр56.
Условие задачи "
                        "\n"
                        "из пр56: Заменить в третьем столбце матрицы A (5×7) все нули на
единицы,\n"
                        "а в пятом столбце матрицы  B (4×5) — все единицы на нули \n")

        # кнопка заполнения матрицы A
        self.gui.btn_generate_matrixA.clicked.connect(self.input_martixA_in_tableA)

        # кнопка заполнения модифицированной матрицы A
        self.gui.btn_modificate_matrixA.clicked.connect(self.input_matrixA_in_modificate_tableA)

        # кнопка заполнения матрицы B
        self.gui.btn_generate_matrixB.clicked.connect(self.input_martixB_in_tableB)

        # кнопка заполнения модифицированной матрицы B
        self.gui.btn_modificate_matrixB.clicked.connect(self.input_matrixB_in_modificate_tableB)


if __name__ == "__main__":
    app = QtWidgets.QApplication([])
    application = CurseProjectWinApp()
    application.show()
    sys.exit(app.exec())
```

2.3.2. Тестирование программы оконного приложения

На рисунке 2.12 представлен скриншот проводимого тестирования программы оконного приложения. Компоненты button1, button2, button3, button4 и button5 с именами: «Генерация матрицы A» - создаем матрицу A(5x7), «Модификация матрицы A» - изменяем матрицу следуя условию задачи, «Генерация матрицы B» - создаем матрицу B(4x5), «Модификация матрицы B» - изменяем матрицу следуя условию задачи, и кнопка с иконкой информации – для отображения, при наведении курсора, условия задачи и кто выполнил работу, пример предоставлен на рисунке 2.13.

По результатам сработанного алгоритма можно увидеть, что мы изменили наши матрицы заменив 1 и 0.

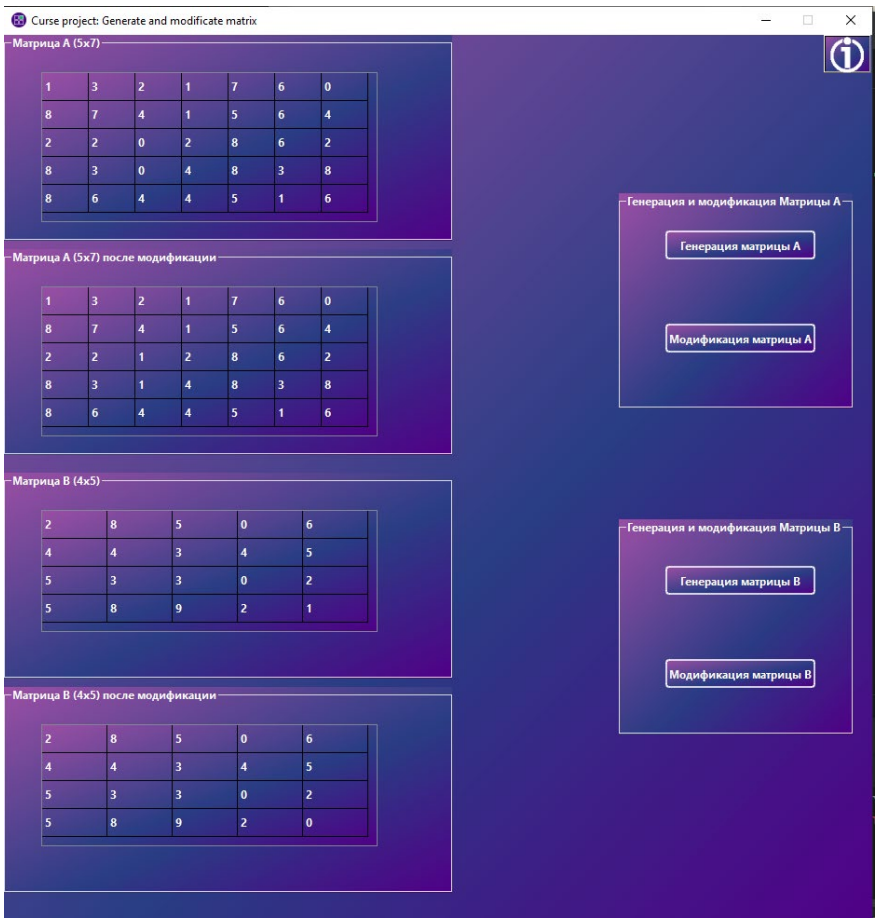

Рисунок 2.12. Результат работы оконной формы программы

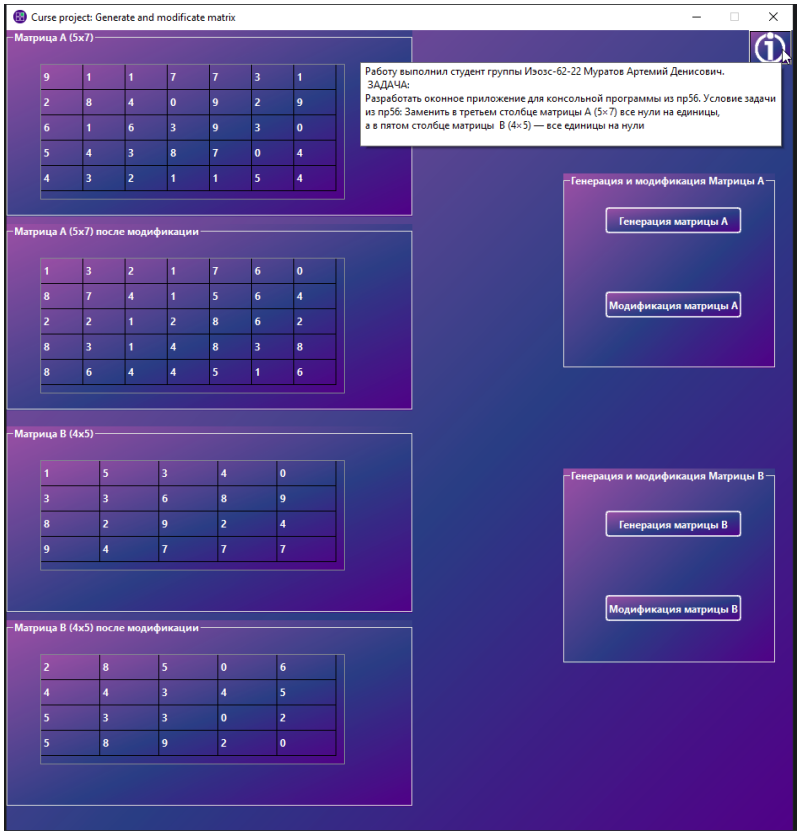

Рисунок 2.13. Вывод информации при наведении на иконку

Вывод

В этой главе были рассмотрены особенности реализации принципа наследования, работы с подпрограммами в алгоритмическом языке Python, а также было разработано оконное приложение. Из этого можно сделать вывод, алгоритм подпрограммы описывает процесс преобразования её входных параметров в результаты, а класс — это план объекта, где вы можете определить, каковы функции объекта, а также свойства объекта.

Заключение

В ходе написания курсовой работы нами были подробно рассмотрены такие UML-диаграммы: диаграмма классов, диаграмма последовательности и диаграмма деятельности. А также были разработаны программы на основе поставленной задачи в алгоритмическом языке Python:

- Консольное приложение с использованием функций;
- Консольное приложение с использованием классов «наследников;
- Оконное приложение QTMainWindow.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений.

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования.

Диаграмма деятельности, нужна для демонстрации рабочего процесса некоторой деятельности, основанной на поэтапных действиях и действиях с поддержкой выбора и параллелизма.

Важной целью языка программирования было облегчение написания больших программ с минимизацией ошибок по сравнению с C++ или C#, следуя принципам процедурного программирования, но избегая всего, что может привести к дополнительным накладным расходам.

В результате проведенной работы показали, что объектно-ориентированный анализ (ООА) – это метод определения важных сущностей реального мира для понимания и объяснения того, как они взаимодействуют между собой. Основное различие между объектно-ориентированным анализом и другими формами анализа заключается в том, что в объектно-

ориентированном подходе требования организованы вокруг объектов, которые объединяют как данные, так и функции. Они моделируются по объектам реального мира, с которыми взаимодействует система. В традиционных методологиях анализа эти два аспекта – функции и данные – рассматриваются отдельно, а объектно-ориентированное программирование позволяет наиболее удобно работать с предметами, имеющими большое число параметров, тогда как использование процедурного подхода в данном случае усложняет написание кода из-за большого числа формальных параметров функций. ООП предоставляет «естественную» декомпозицию, позволяя вести независимую разработку отдельных частей программы, а использование принципов инкапсуляции, наследования и полиморфизма позволяет конструировать более сложные классы из сравнительно простых.

Список литературы

1. В.С. Зубов, В.С. Батасова. Сборник задач по базовой компьютерной подготовке: учебное пособие по курсу «Информатика». – М.: Издательский дом МЭИ, 2007. – 124 с.
2. Лекция 1 «Введение в объектно-ориентированное программирование».
3. Лекция 2 «Принципы объектно-ориентированного программирования».
4. Python 3 и PyQt 6. Разработка приложений. Автор: Прохоренок Н. А., Дронов В. А. Дата выхода: 2023. Издательство: «БХВ-Петербург». Количество страниц: 834