**Отчет по рубежному контролю №2 по курсу**

**базовые компоненты интернет-технологий (БКИТ)**

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-54Б Папин Алексей | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Гапанюк Ю.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Москва, МГТУ - 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Листинг программы: 2](#_Toc120742435)

[1.1. RK\_2.py (Основной файл) 2](#_Toc120742436)

[1.2. test.py (Файл модульного теста) 5](#_Toc120742437)

[2. Результаты работы программы: 7](#_Toc120742438)

[2.1. Основной файл (RK\_2.py) 7](#_Toc120742439)

[2.2. Модульный тест-файл (test.py) 7](#_Toc120742440)

[2.2.1.В IDE JetBrains PyCharm 7](#_Toc120742441)

[2.2.2.В cmd / powershell 8](#_Toc120742442)

# **Листинг программы:**

# **RK\_2.py (Основной файл)**

from operator import itemgetter  
  
'''  
Вариант Г.  
«Книга» и «Глава» связаны соотношением один-ко-многим.   
Выведите список всех книг, у которых название начинается с буквы «А», и список содержащих в них глав.  
  
«Книга» и «Глава» связаны соотношением один-ко-многим.   
Выведите список книг с максимальной страницы глав в каждой книге, отсортированный по максимальной странице.  
  
«Книга» и «Глава» связаны соотношением многие-ко-многим.   
Выведите список всех связанных глав и книг, отсортированный по книгам, сортировка по главам произвольная.   
'''  
  
  
# класс Глава  
class Chapter:  
 def \_\_init\_\_(self, id, name, page, book\_id):  
 # номер главы  
 self.id = id  
 # название главы  
 self.name = name  
 # страница на которой находится глава  
 self.page = page  
 # номер книги  
 self.book\_id = book\_id  
  
  
# класс книга  
class Book:  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class ChapterBook:  
 def \_\_init\_\_(self, book\_id, chapter\_id):  
 self.book\_id = book\_id  
 self.chapter\_id = chapter\_id  
  
  
books = [  
 Book(1, 'СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ.'),  
 Book(2, 'Теоретические основы электротехники. Электрические цепи.'),  
 Book(3, 'Английский язык (Для технических университетов и вузов)'),  
 Book(4, 'Руководство для начинающих С++. Второе издание'),  
 Book(5, 'Архитектура АСОИУ. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ')  
]  
chapters = [  
 Chapter(1, 'Основные положения теории электромагнитного поля и их применение к теории электрических цепей', 8, 2),  
 Chapter(2, 'ОПИСАНИЕ СХЕМ', 23, 1),  
 Chapter(3, 'Основы С++', 22, 4),  
 Chapter(4, 'ОПИСАНИЕ СИМВОЛОВ', 139, 1),  
 Chapter(5, 'Не линейные электрические цепи временного тока', 70, 2),  
 Chapter(6, 'АРХИЕКТУРНОЕ ПОНЯТИЕ МНОЖЕСТВА', 4, 5)  
]  
chapters\_of\_books = [  
 ChapterBook(2, 1),  
 ChapterBook(1, 2),  
 ChapterBook(4, 3),  
 ChapterBook(1, 4),  
 ChapterBook(2, 5),  
 ChapterBook(5, 6),  
]  
  
  
def one\_to\_many(books, chapters):  
 return [(ch.name, ch.page, book.name)  
 for book in books  
 for ch in chapters  
 if ch.book\_id == book.id]  
  
  
def many\_to\_many\_temp(books, chapters\_of\_books):  
 return [(book.name, ChOfBooks.book\_id, ChOfBooks.chapter\_id)  
 for book in books  
 for ChOfBooks in chapters\_of\_books  
 if book.id == ChOfBooks.book\_id]  
  
  
def many\_to\_many(books, chapters):  
 return [(ch.name, ch.page, book\_name)  
 for book\_name, book\_id, ch\_id in many\_to\_many\_temp(books, chapters\_of\_books)  
 for ch in chapters if ch.id == ch\_id]  
  
  
def exercise\_G1(books, chapters):  
 array\_dict = {}  
 for lib\_name, x, book\_name in one\_to\_many(books, chapters):  
 # если название книг начинается с 'А'  
 if book\_name[0] == 'А':  
 if book\_name in array\_dict:  
 array\_dict[book\_name].append(lib\_name)  
 else:  
 array\_dict[book\_name] = [lib\_name]  
  
 return array\_dict.items()  
  
def exercies\_G2(books, chapters):  
 array\_dict\_2 = {}  
 for x, func\_num, book\_name in one\_to\_many(books, chapters):  
 if book\_name in array\_dict\_2:  
 array\_dict\_2[book\_name] = max(array\_dict\_2[book\_name], func\_num)  
 else:  
 array\_dict\_2[book\_name] = func\_num  
 array\_dict\_2 = {key: value for key, value in sorted(array\_dict\_2.items(), key=lambda item: item[1])}  
 return array\_dict\_2.items()  
  
def exercise\_G3(books, chapters):  
 array\_list = []  
 for lib\_name, x, book\_name in many\_to\_many(books, chapters):  
 array\_list.append((book\_name, lib\_name))  
 array\_list = sorted(array\_list, key=lambda item: item[0])  
 return array\_list  
  
def main():  
 print('Задание Г1')  
 print(\*exercise\_G1(books, chapters))  
  
 print('Задание Г2')  
 print(\*exercies\_G2(books, chapters))  
  
 print('Задание Г3')  
 print(\*exercise\_G3(books, chapters))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# **test.py (Файл модульного теста)**

import unittest  
  
from RK\_1 import Chapter, Book, ChapterBook, exercise\_G1, exercies\_G2, exercise\_G3, books, chapters, chapters\_of\_books  
  
simple\_book = [  
 Book(1, 'Азбука'),  
 Book(2, 'Программирование на Python'),  
 Book(3, 'Математика')  
]  
simple\_chapter = [  
 Chapter(1, 'Модульный тест', 10, 2),  
 Chapter(2, 'Буква А', 1, 1),  
 Chapter(3, 'Плюсик', 4, 3)  
]  
simple\_chapter\_of\_books = [  
 ChapterBook(1, 2),  
 ChapterBook(2, 1),  
 ChapterBook(3, 3)  
]  
  
# Тест на создание класс "Глава"  
class test\_programms(unittest.TestCase):  
  
 # Класс Chapter с пустыми параметрами  
 def test\_empty\_parametrs\_class\_chapter(self):  
 with self.assertRaises(TypeError) as context:  
 # Выводит ошибку, т.к. в классе не было инициализировано по умолчанию  
 Chapter()  
  
 self.assertEqual(  
 "\_\_init\_\_() missing 4 required positional arguments: 'id', 'name', 'page', " "and 'book\_id'",  
 str(context.exception))  
  
 # Класс Chapter с пустыми значениями  
 def test\_empty\_class\_chapter(self):  
 myclass\_chapter = Chapter(None, None, None, None)  
 self.assertEqual(myclass\_chapter.id, None)  
 self.assertEqual(myclass\_chapter.name, None)  
 self.assertEqual(myclass\_chapter.page, None)  
 self.assertEqual(myclass\_chapter.book\_id, None)  
  
 # Класс Chapter с значением  
 def test\_class\_chapter(self):  
 myclass\_chapter = Chapter(1, 'Математическое моделирование', 8, 2)  
 self.assertEqual(myclass\_chapter.id, 1)  
 self.assertEqual(myclass\_chapter.name, 'Математическое моделирование')  
 self.assertEqual(myclass\_chapter.page, 8)  
 self.assertEqual(myclass\_chapter.book\_id, 2)  
  
 # Класс Book с значением  
 def test\_class\_book(self):  
 myclass\_book = Book(1, 'СХЕМЫ АЛГОРИТМОВ')  
 self.assertEqual(myclass\_book.id, 1)  
 self.assertEqual(myclass\_book.name, 'СХЕМЫ АЛГОРИТМОВ')  
  
 # Класс ChapterBook с значением  
 def test\_class\_chapter\_book(self):  
 myclass\_chapter\_book = ChapterBook(2, 1)  
 self.assertEqual(myclass\_chapter\_book.book\_id, 2)  
 self.assertEqual(myclass\_chapter\_book.chapter\_id, 1)  
  
 # Проверка задания Г1  
 def test\_exercise\_G1(self):  
 self.assertEqual(dict(exercise\_G1(simple\_book, simple\_chapter)), {'Азбука': ['Буква А']})  
  
 # Проверка задания Г2  
 def test\_exercise\_G2(self):  
 self.assertEqual(dict(exercies\_G2(simple\_book, simple\_chapter)), {'Азбука': 1, 'Математика': 4, 'Программирование на Python': 10})  
  
 # Проверка задания Г3  
 def test\_exercise\_G3(self):  
 self.assertEqual(exercise\_G3(simple\_book, simple\_chapter), [('Азбука', 'Буква А'), ('Программирование на Python', 'Модульный тест')])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

# **Результаты работы программы:**

# **Основной файл (RK\_2.py)**

Задание Г1

('Архитектура АСОИУ. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ', ['АРХИЕКТУРНОЕ ПОНЯТИЕ МНОЖЕСТВА'])

Задание Г2

('Архитектура АСОИУ. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ', 4) ('Руководство для начинающих С++. Второе издание', 22) ('Теоретические основы электротехники. Электрические цепи.', 70) ('СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ.', 139)

Задание Г3

('Архитектура АСОИУ. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ', 'АРХИЕКТУРНОЕ ПОНЯТИЕ МНОЖЕСТВА') ('Руководство для начинающих С++. Второе издание', 'Основы С++') ('СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ.', 'ОПИСАНИЕ СХЕМ') ('СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ, ДАННЫХ И СИСТЕМ.', 'ОПИСАНИЕ СИМВОЛОВ') ('Теоретические основы электротехники. Электрические цепи.', 'Основные положения теории электромагнитного поля и их применение к теории электрических цепей') ('Теоретические основы электротехники. Электрические цепи.', 'Не линейные электрические цепи временного тока')

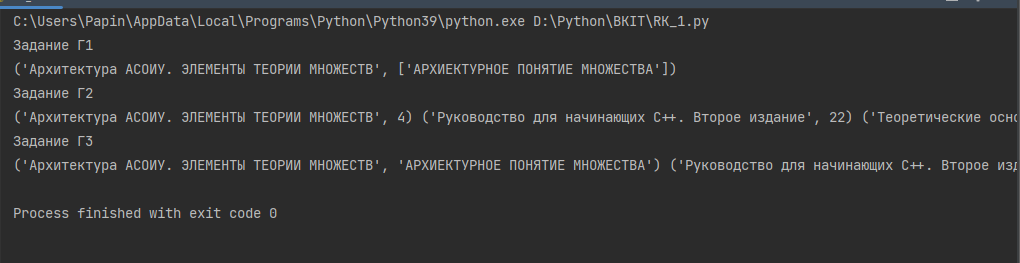
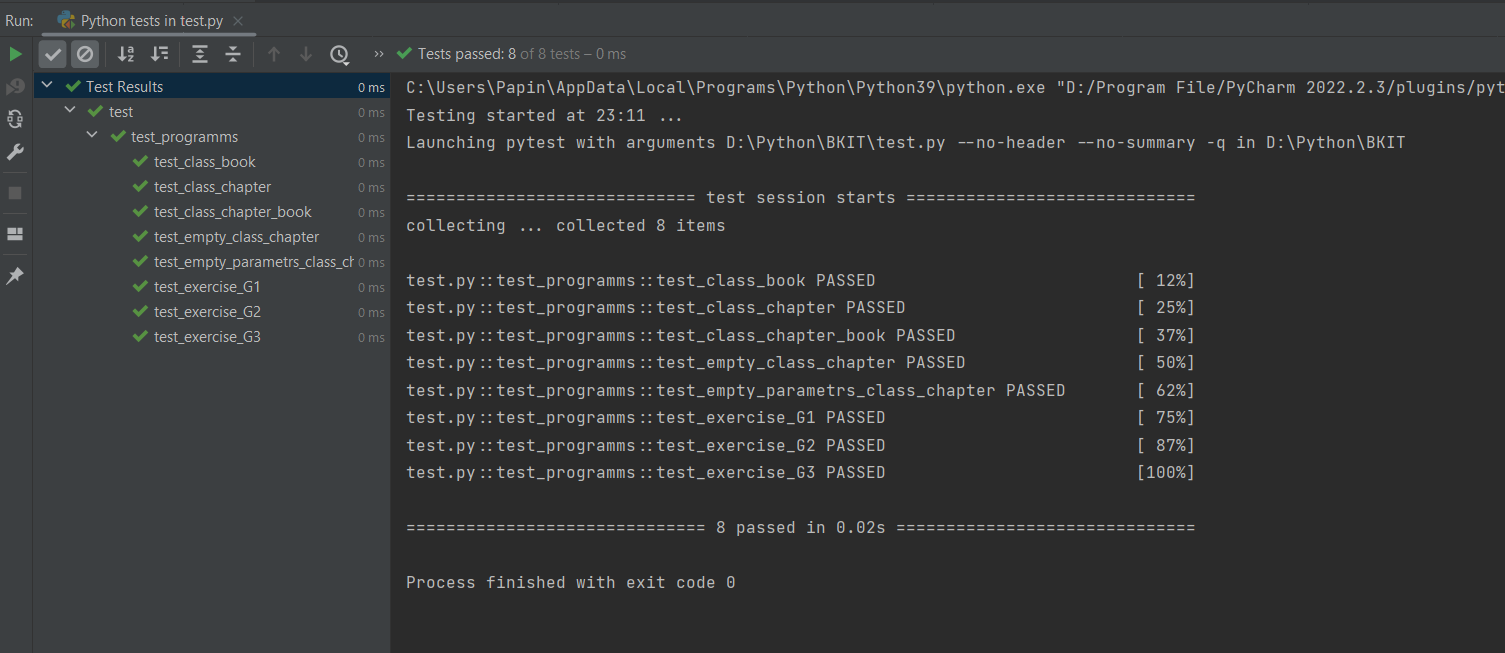


Рисунок 1 - Скриншот результатов работы программы

# **Модульный тест-файл (test.py)**

# **В IDE JetBrains PyCharm**



# **В cmd / powershell**

