|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №1  по дисциплине «Языки программирования»  Вариант 10 | | |
|  | | Работу выполнил:  студент группы ИТ/О, ИТ-6-2025 НБ  1 курса Рисков Г. С.  «7» января 2024 г. |
| Работу проверил:  Рубцова М.Б.  «15» января 2024 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc187790863)

[Алгоритм решения 4](#_Toc187790864)

[Тестирование 10](#_Toc187790865)

[Код программы 11](#_Toc187790866)

# Постановка задачи

Создать файл записей, в котором хранится информация о дисциплинах, читаемых на факультете: название, с какого семестра читается, продолжительность курса (в семестрах), общее количество часов, вид отчётности (зачёт, экзамен), читающая курс кафедра Разработать и реализовать программу "Учебный план", которая считывает исходную информацию и позволяет на основе неё создавать следующие отчёты: 1 Полный список всех дисциплин, который будет отсортирован следующему ключу: с какого семестра читается (по возрастанию) + читающая кафедра (по возрастанию) + общее количество часов (по убыванию). 2 Список всех дисциплин с заданным видом отчётности, отсортированный по следующему ключу: продолжительность курса (по возрастанию) + общее количество часов (по убыванию). 3 Список всех дисциплин с общим количеством часов от N1 до N2, отсортированный по следующему ключу: читающая кафедра (по возрастанию) + общее количество часов 9по убыванию). Создать базу дисциплин, включающую не менее 25 записей и на основе неё сформировать все указанные списки. База должна содержать такие записи, чтобы во всех списках явно прослеживался заданный вид сортировки по всем ключам. Для сортировки записей использовать сортировку бинарными вставками.

# 

# Алгоритм решения

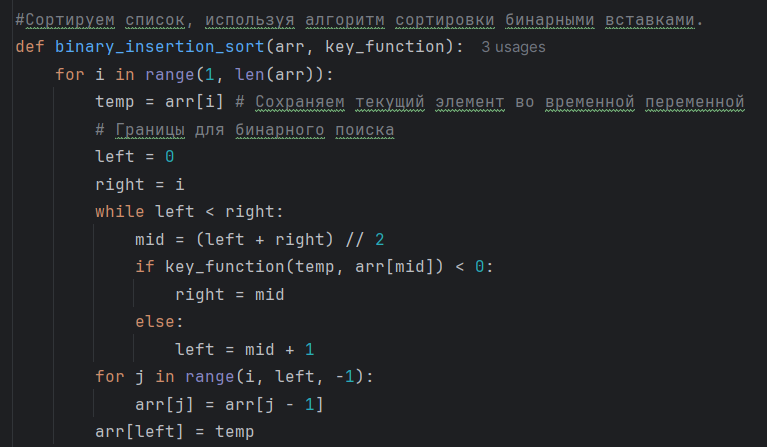
1. Создание txt файла с данными о занятиях

Математический анализ;1;2;180;экзамен;Кафедра математики  
Линейная алгебра;1;1;100;зачёт;Кафедра математики  
Дискретная математика;2;1;120;зачёт;Кафедра информатики  
Основы программирования;2;2;200;экзамен;Кафедра информатики  
Базы данных;3;1;150;экзамен;Кафедра информационных технологий  
Операционные системы;3;1;110;зачёт;Кафедра информационных технологий  
Физика;1;2;150;экзамен;Кафедра физики  
Химия;1;1;80;зачёт;Кафедра химии  
История;2;1;90;зачёт;Кафедра истории  
Экономика;2;2;200;экзамен;Кафедра экономики  
Философия;3;1;70;зачёт;Кафедра философии  
Теория вероятностей;2;1;120;экзамен;Кафедра математики  
Методы оптимизации;3;2;180;экзамен;Кафедра информатики  
Алгоритмы и структуры данных;3;2;220;экзамен;Кафедра информатики  
Веб-технологии;4;1;120;зачёт;Кафедра информационных технологий  
Мобильная разработка;4;2;180;экзамен;Кафедра информационных технологий  
Английский язык;1;2;100;зачёт;Кафедра иностранных языков  
Русский язык;1;1;60;зачёт;Кафедра русского языка  
Культурология;2;1;70;зачёт;Кафедра культурологии  
Социология;3;1;80;зачёт;Кафедра социологии  
Политология;3;1;90;экзамен;Кафедра политологии  
Правоведение;4;1;100;зачёт;Кафедра правоведения  
Психология;4;2;120;зачёт;Кафедра психологии  
Искусственный интеллект;4;2;250;экзамен;Кафедра информатики  
Компьютерная графика;3;2;220;зачёт;Кафедра информационных технологий

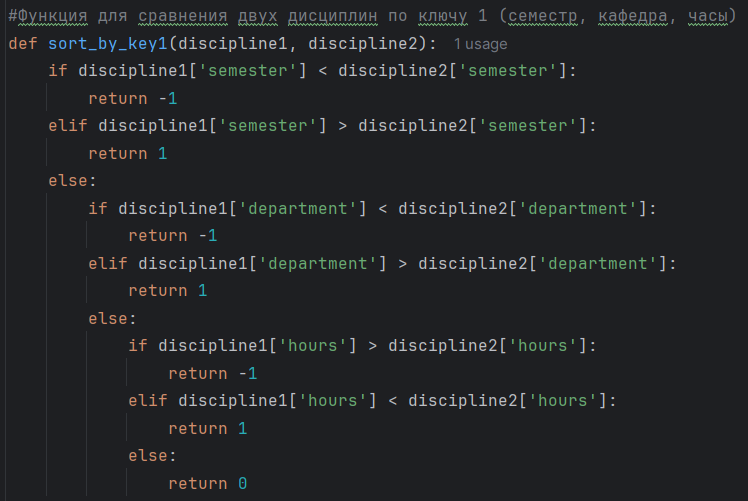
* + Создаю функцию, которая будет читать данные о дисциплинах из файла и возвращать их в виде списка словарей.
  + Добавил альтернативный вывод при условии, если путь до файла указан неправильно, либо возникла другая ошибка.



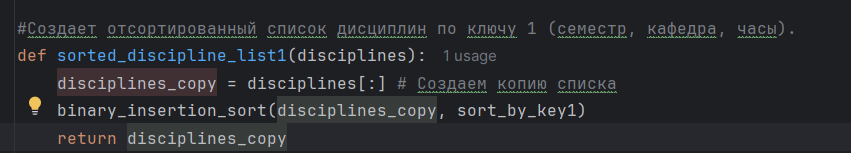
1. Создаю функцию, которая сортирует список data с помощью алгоритма бинарных вставок, используя функцию key\_func для сравнения элементов.



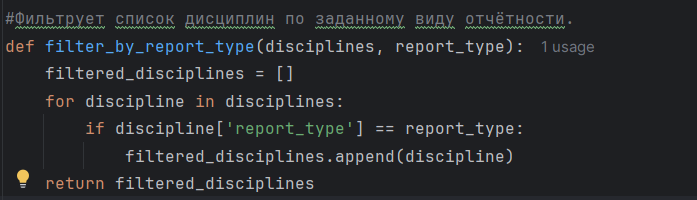
1. Создаю Функцию sort\_by\_key1 предназначена для использования в алгоритмах сортировки, чтобы определить порядок дисциплин. Она позволяет сортировать список дисциплин сначала по семестрам (по возрастанию), затем, внутри каждого семестра, по кафедре (в алфавитном порядке) и, наконец, по количеству часов (по убыванию).



1. Создаю функцию sorted\_discipline\_list1,которая принимает список дисциплин, создаёт его копию, сортирует эту копию по семестру, кафедре и часам (с использованием бинарной сортировки вставками и функции сравнения sort\_by\_key1) и возвращает отсортированную копию, не изменяя исходный список.

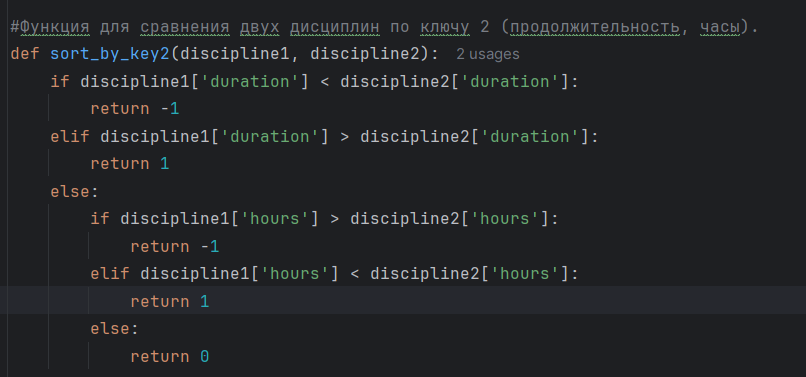


1. Создаю функцию filter\_by\_report\_type, которая принимает список дисциплин и тип отчётности, и возвращает новый список, содержащий только те дисциплины, у которых тип отчётности совпадает с заданным.

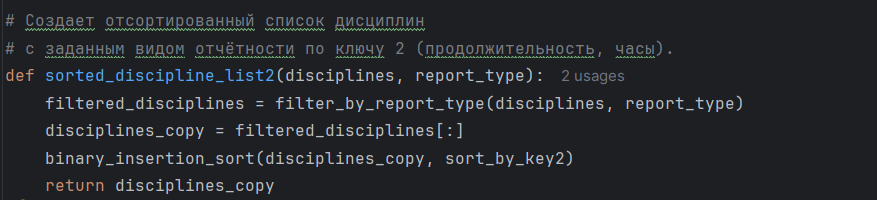


1. Создаю Функцию sort\_by\_key2, которая сравнивает две дисциплины (discipline1 и discipline2) по продолжительности (duration) и количеству часов (hours). Она возвращает:

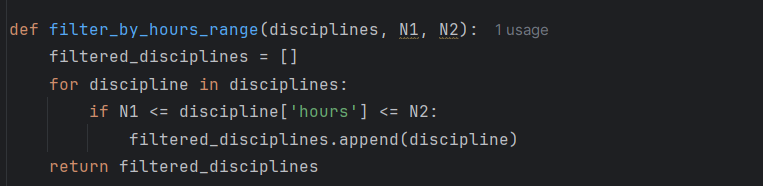
* -1, если discipline1 меньше discipline2 по заданному критерию.
* 1, если discipline1 больше discipline2 по заданному критерию.
* 0, если discipline1 и discipline2 равны по заданному критерию.



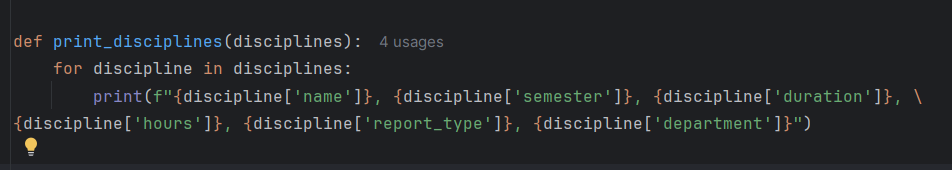
1. Создаю функцию sorted\_discipline\_list2, которая фильтрует список дисциплин по заданному типу отчётности, создаёт копию отфильтрованного списка, сортирует эту копию по продолжительности (сначала меньшая) и часам (сначала больше) и возвращает отсортированный список.



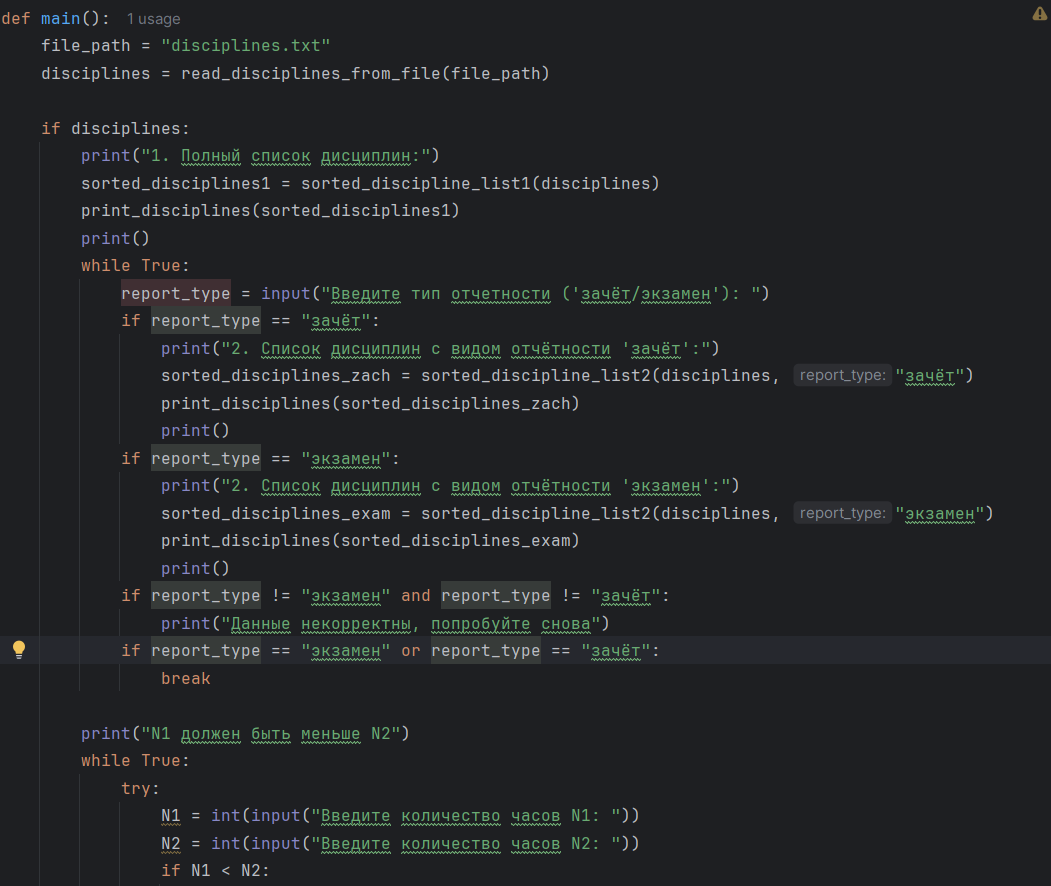
1. Создаю функцию filter\_by\_hours\_range, которая принимает список дисциплин и два числа N1 и N2, и возвращает новый список, содержащий только те дисциплины, у которых количество часов (hours) находится в диапазоне от N1 до N2 включительно.



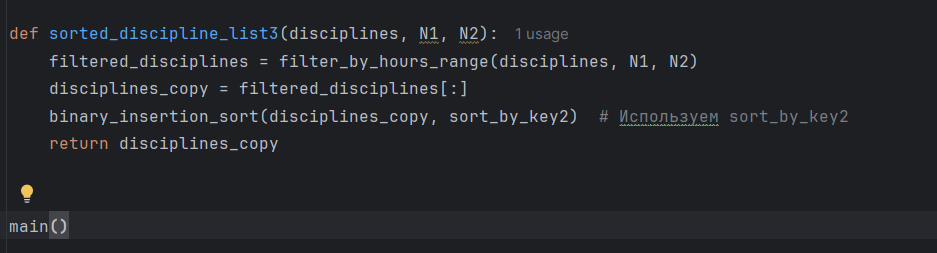
1. Функцию print\_disciplines, которая выводит имя, семестр, продолжительность, количество часов, тип отчета и кафедру каждой дисциплины.



1. Создаю функцию main, которая выполняет следующие действия:
   * Читает данные: Загружает список дисциплин из файла “disciplines.txt”.
   * Выводит полный список: Выводит отсортированный по семестру, кафедре и часам полный список дисциплин.
   * Запрашивает тип отчётности: Запрашивает у пользователя тип отчётности (“зачёт” или “экзамен”).
   * Выводит отфильтрованный список: Выводит отсортированный по продолжительности и часам список дисциплин, соответствующих введенному типу отчётности.
   * Запрашивает диапазон часов: Запрашивает у пользователя два числа, N1 и N2, представляющие диапазон часов.
   * Выводит отфильтрованный список по диапазону: Выводит отфильтрованный список дисциплин, количество часов которых попадает в заданный диапазон (от N1 до N2).

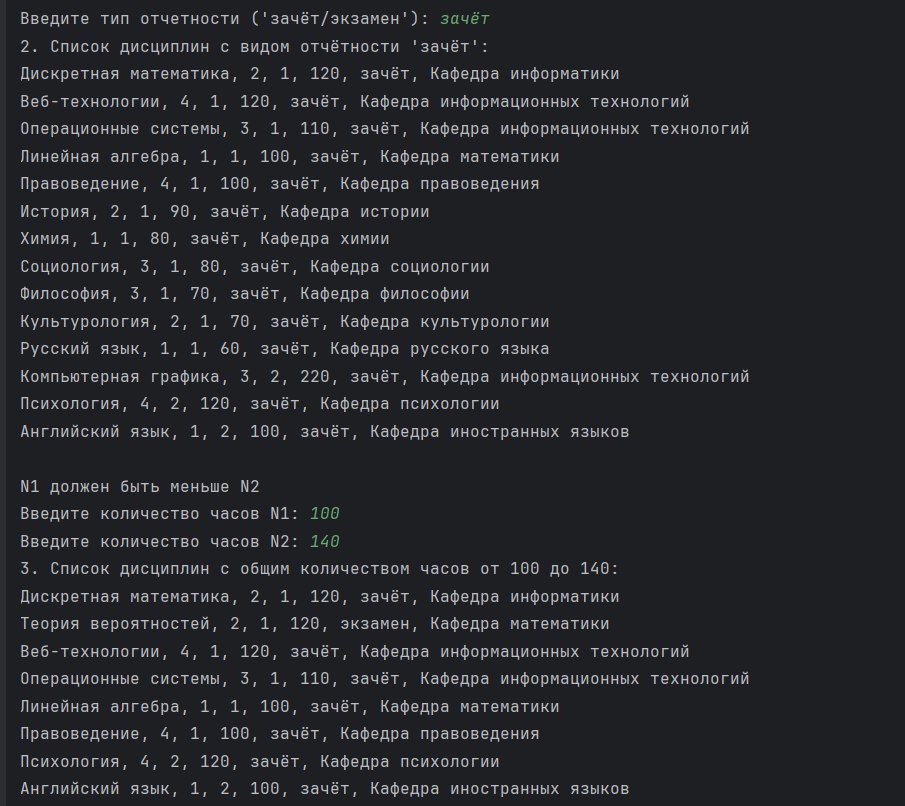


1. Создаю функцию sorted\_discipline\_list3 фильтрует список дисциплин по заданному диапазону часов (от N1 до N2), создаёт копию отфильтрованного списка, сортирует эту копию по продолжительности (сначала меньшая) и часам (сначала больше) и возвращает отсортированный список. Вывожу main()

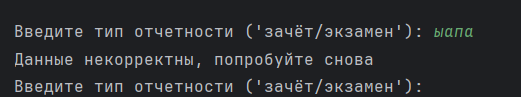


# Тестирование

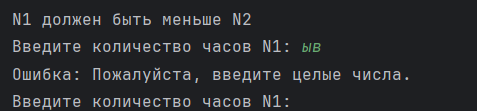
При правильном вводе



При неправильном вводе отчётности



При неправильном вводе часов



# Код программы

def read\_disciplines\_from\_file(file\_path):  
 disciplines = [] # Пустой массив для хранения дисциплин  
 try:  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 for line in file:  
 line = line.strip() #Удаляем пробелы  
 if line:  
 data = line.split(';')  
 if len(data) == 6:  
 discipline = {  
 'name': data[0],  
 'semester': int(data[1]),  
 'duration': int(data[2]),  
 'hours': int(data[3]),  
 'report\_type': data[4],  
 'department': data[5]  
 }  
 disciplines.append(discipline)  
 except FileNotFoundError: # Если файл не найден  
 print("Файл не найден")  
 return []  
 return disciplines  
  
  
#Сортируем список, используя алгоритм сортировки бинарными вставками.  
def binary\_insertion\_sort(arr, key\_function):  
 for i in range(1, len(arr)):  
 temp = arr[i] # Сохраняем текущий элемент во временной переменной  
 # Границы для бинарного поиска  
 left = 0  
 right = i  
 while left < right:  
 mid = (left + right) // 2  
 if key\_function(temp, arr[mid]) < 0:  
 right = mid  
 else:  
 left = mid + 1  
 for j in range(i, left, -1):  
 arr[j] = arr[j - 1]  
 arr[left] = temp  
  
#Функция для сравнения двух дисциплин по ключу 1 (семестр, кафедра, часы)  
def sort\_by\_key1(discipline1, discipline2):  
 if discipline1['semester'] < discipline2['semester']:  
 return -1  
 elif discipline1['semester'] > discipline2['semester']:  
 return 1  
 else:  
 if discipline1['department'] < discipline2['department']:  
 return -1  
 elif discipline1['department'] > discipline2['department']:  
 return 1  
 else:  
 if discipline1['hours'] > discipline2['hours']:  
 return -1  
 elif discipline1['hours'] < discipline2['hours']:  
 return 1  
 else:  
 return 0  
  
  
#Создает отсортированный список дисциплин по ключу 1 (семестр, кафедра, часы).  
def sorted\_discipline\_list1(disciplines):  
 disciplines\_copy = disciplines[:] # Создаем копию списка  
 binary\_insertion\_sort(disciplines\_copy, sort\_by\_key1)  
 return disciplines\_copy  
  
  
#Фильтрует список дисциплин по заданному виду отчётности.  
def filter\_by\_report\_type(disciplines, report\_type):  
 filtered\_disciplines = []  
 for discipline in disciplines:  
 if discipline['report\_type'] == report\_type:  
 filtered\_disciplines.append(discipline)  
 return filtered\_disciplines  
  
  
#Функция для сравнения двух дисциплин по ключу 2 (продолжительность, часы).  
def sort\_by\_key2(discipline1, discipline2):  
 if discipline1['duration'] < discipline2['duration']:  
 return -1  
 elif discipline1['duration'] > discipline2['duration']:  
 return 1  
 else:  
 if discipline1['hours'] > discipline2['hours']:  
 return -1  
 elif discipline1['hours'] < discipline2['hours']:  
 return 1  
 else:  
 return 0  
  
# Создает отсортированный список дисциплин  
# с заданным видом отчётности по ключу 2 (продолжительность, часы).  
def sorted\_discipline\_list2(disciplines, report\_type):  
 filtered\_disciplines = filter\_by\_report\_type(disciplines, report\_type)  
 disciplines\_copy = filtered\_disciplines[:]  
 binary\_insertion\_sort(disciplines\_copy, sort\_by\_key2)  
 return disciplines\_copy  
  
  
def filter\_by\_hours\_range(disciplines, N1, N2):  
 filtered\_disciplines = []  
 for discipline in disciplines:  
 if N1 <= discipline['hours'] <= N2:  
 filtered\_disciplines.append(discipline)  
 return filtered\_disciplines  
  
  
def print\_disciplines(disciplines):  
 for discipline in disciplines:  
 print(f"{discipline['name']}, {discipline['semester']}, {discipline['duration']}, \  
{discipline['hours']}, {discipline['report\_type']}, {discipline['department']}")  
  
  
def main():  
 file\_path = "disciplines.txt"  
 disciplines = read\_disciplines\_from\_file(file\_path)  
  
 if disciplines:  
 print("1. Полный список дисциплин:")  
 sorted\_disciplines1 = sorted\_discipline\_list1(disciplines)  
 print\_disciplines(sorted\_disciplines1)  
 print()  
 while True:  
 report\_type = input("Введите тип отчетности ('зачёт/экзамен'): ")  
 if report\_type == "зачёт":  
 print("2. Список дисциплин с видом отчётности 'зачёт':")  
 sorted\_disciplines\_zach = sorted\_discipline\_list2(disciplines, "зачёт")  
 print\_disciplines(sorted\_disciplines\_zach)  
 print()  
 if report\_type == "экзамен":  
 print("2. Список дисциплин с видом отчётности 'экзамен':")  
 sorted\_disciplines\_exam = sorted\_discipline\_list2(disciplines, "экзамен")  
 print\_disciplines(sorted\_disciplines\_exam)  
 print()  
 if report\_type != "экзамен" and report\_type != "зачёт":  
 print("Данные некорректны, попробуйте снова")  
 if report\_type == "экзамен" or report\_type == "зачёт":  
 break  
  
 print("N1 должен быть меньше N2")  
 while True:  
 try:  
 N1 = int(input("Введите количество часов N1: "))  
 N2 = int(input("Введите количество часов N2: "))  
 if N1 < N2:  
 print(f"3. Список дисциплин с общим количеством часов от {N1} до {N2}:")  
 sorted\_disciplines3 = sorted\_discipline\_list3(disciplines, N1, N2)  
 print\_disciplines(sorted\_disciplines3)  
 print()  
 break  
 else:  
 print("N1 должно быть меньше N2. Попробуйте снова.")  
 except ValueError:  
 print("Ошибка: Пожалуйста, введите целые числа.")  
  
  
  
def sorted\_discipline\_list3(disciplines, N1, N2):  
 filtered\_disciplines = filter\_by\_hours\_range(disciplines, N1, N2)  
 disciplines\_copy = filtered\_disciplines[:]  
 binary\_insertion\_sort(disciplines\_copy, sort\_by\_key2) # Используем sort\_by\_key2  
 return disciplines\_copy  
  
  
main()