



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Новосибирский государственный университет экономики и управления «НИНХ»  
(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

**Кафедра информационных технологий**

## **КУРСОВАЯ РАБОТА**

**Программное приложение «Каталог звезд нашей солнечной системы»**

Дисциплина: Программирование

Ф.И.О студента: Ярославцев Никита Витальевич

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Программная инженерия

Номер группы: ФИ202

Номер зачетной книжки: 220145

Проверил: Пестунов Андрей Игоревич, Кандидат физико-математических наук,  
Заведующий кафедрой информационных технологий

Новосибирск 2023



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»  
(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

**Кафедра информационных технологий**

**ЗАДАНИЕ  
на курсовую работу**

Тема \_\_\_\_\_ Программное приложение «Каталог звезд нашей солнечной системы» \_\_\_\_\_

ФИО студента \_\_\_\_\_ Ярославцев Никита Витальевич \_\_\_\_\_

Группа \_\_\_\_\_ ФИ202 \_\_\_\_\_

Перечень подлежащих разработке вопросов и календарный график

№ п/п	Наименование вопросов, подлежащих разработке (этапы работы)	Срок выполнения
1.	Ознакомление с заданием, уточнение требований	31.03.2023
1.	Разработка и тестирование программного приложения согласно варианту	05.04.2023
2	Создание скриншотов с демонстрацией работы программы и компоновка GIF-файла	10.04.2023
3	Оформление текста курсовой работы согласно стандарту	10.04.2023
4	Прикрепление текста курсовой работы, GIF-файла и программного кода в электронный курс	20.04.2023
5	Защита текста курсовой работы преподавателю	30.04.2023

Дата выдачи задания «\_\_\_» \_\_\_\_\_ 20\_\_\_ года

Срок сдачи работы «\_\_\_» \_\_\_\_\_ 20\_\_\_ года

Преподаватель \_\_\_\_\_

(фамилия и инициалы преподавателя)

\_\_\_\_\_  
(подпись)

Задание получил студент \_\_\_\_\_

(фамилия и инициалы студента)

\_\_\_\_\_  
(подпись)



**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение  
высшего образования**

**«Новосибирский государственный университет экономики и управления «НИНХ»  
(ФГБОУ ВО «НГУЭУ», НГУЭУ)**

Кафедра информационных технологий

### **ЗАЯВЛЕНИЕ**

**о самостоятельном характере выполненной работы**

Я, \_\_\_\_\_ Ярославцев \_ Никита \_ Витальевич \_\_\_\_\_,  
студент группы \_\_\_\_\_ ФИ202\_\_\_\_, направления подготовки \_\_\_\_\_  
02.03.02 Фундаментальная информатика и информационные технологии\_\_\_\_,  
направленности (профиля) \_\_\_\_\_ программная инженерия\_\_\_\_\_,  
заявляю, что в моей курсовой работе, выполненной на тему:  
Программное приложение «Каталог звезд нашей солнечной системы» \_\_\_\_\_,  
не содержится элементов плагиата.

Все заимствования из печатных и электронных источников, а также из  
защищенных ранее письменных работ, кандидатских и докторских  
диссертаций имеют соответствующие ссылки.

«10» апреля 2023 г.

\_\_\_\_\_  
(подпись) И.О. Фамилия

Результаты проверки в системе «Антиплагиат»

Доля авторского текста (оригинальности) в результате автоматизированной  
проверки составила \_\_\_\_\_98\_\_\_\_\_ %.

Руководитель курсовой работы \_\_\_\_ кандидат физико-математических наук, зав.  
каф. информационных технологий, Пестунов А. И. \_\_\_\_\_,  
(уч. степень, должность, Фамилия И.О.)

« \_\_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

\_\_\_\_\_  
(подпись)

# СОДЕРЖАНИЕ

<b>1. РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ</b>	<b>5</b>
1.1. ЗАПУСК ПРИЛОЖЕНИЯ	5
1.2. ОБРАБОТКА КОМАНД ОТ ПОЛЬЗОВАТЕЛЯ	6
1.3. КОМАНДЫ «ПОМОЩЬ» И «ВЫХОД»	7
1.4. КОМАНДА «ДОБАВИТЬ»	8
1.5. КОМАНДА «УДАЛИТЬ»	10
1.6. КОМАНДА «СПИСОК»	11
1.7. КОМАНДА «СОРТИРОВАТЬ»	13
1.8. КОМАНДА «СОРТИРОВАТЬ»	13
1.9. КОМАНДА «ПЕРЕВОД»	16
1.10. КОМАНДА «ФИЛЬТР»	17
<b>2. СПИСОК ТЕСТОВЫХ СЛУЧАЕВ ДЛЯ РУЧНОГО ТЕСТИРОВАНИЯ ПРОГРАММЫ</b>	<b>18</b>
<b>3. СПИСОК ИСПОЛЪЗУЕМОЙ ЛИТЕАТУРЫ</b>	<b>20</b>

# 1. РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Поставленной задачей является, написание консольного приложения на языке Python [1], которое должно выполнять следующие функции:

1. Вывод всех объектов в виде таблицы.
2. Выход из программы.
3. Добавление нового объекта (значения свойств ввести с клавиатуры).
4. Удаление выбранного пользователем объекта.
5. Сортировка объектов по выбранному пользователем свойству.
6. Фильтрация (вывод объектов, у которых масса не превосходит заданной величины).
7. Преобразование значения свойства у всех объектов (вывод расстояния до всех звезд в парсеках вместо световых лет).

Консольное приложение или программа командной строки - это компьютерная программа, предназначенная для использования через текстовый пользовательский интерфейс, такой как текстовый терминал, интерфейс командной строки некоторых операционных систем (Unix, DOS, и т.д.) или текстовый интерфейс, входящий в состав большинства операционных систем с графическим пользовательским интерфейсом (GUI), таких как консоль Windows в Microsoft Windows [2], Terminal в macOS [3] и xterm в Unix [4].

## 1.1. Запуск приложения

Исходя из задания, программа должна выполнять следующие команды: «добавить», «удалить», «список», «сортировать», «фильтр», «перевод», «выход».

Листинг 1 – Вывод списка команд и приглашение на взаимодействие

```
if __name__ == "__main__":  
    print("«Каталог звезд нашей галактики»\n\nСписок возможных  
команд:\n"  
        "добавить - добавление звезды\n"  
        "удалить - удаление выбранной звезды\n"  
        "список - вывод всех звезд в виде таблицы\n"  
        "сортировать - сортировка звезд по выбранному параметру\n"  
        "фильтр - вывод всех звезд, у которых масса не превосходит  
заданной величины\n")
```

```

        "перевод - вывод расстояния от Солнца до звезд в парсеках\n"
        "выход - выход из программы\n")
while True:
    command = input("Введите команду: ")
    executing_the_command(command)

```

Обратимся к листингу 1.

Если мы захотим импортировать наши процедуры в другие проекты, то конструкция `if __name__ == "__main__":` не позволит перенести в новый проект, часть активного кода, работа которого не будет полезна. Далее с помощью `print("")` для человека единожды выводится название программы и список возможных команд с кратким пояснением их работы. Последние три строки листинга самые важные, здесь запускается вечный цикл, приглашающий пользователя ввести команду и исполнение этой команды с помощью процедуры `executing_the_command()` (листинг 2.).

## 1.2. Обработка команд от пользователя

### Листинг 2 – Выполнение команды

```

def executing_the_command():
    try:
        commands = {"выход": close_console, "добавить": add_obj,
                    "удалить": del_obj, "список": list_obj,
                    "сортировать": sort_obj, "фильтр": filter_obj,
                    "перевод": convert_obj}
        commands[command.lower().split(" ")[0]]()
    except KeyError:
        print("Такой команды не существует!")

```

Рассмотрим в конструкции `commands[command.lower().split(" ")[0]]()` (листинг 2). Данная строка обращается к словарю `commands`, по ключу-команде создает ссылку на процедуру ответственную за выполнение запрашиваемых действий и инициализируется с помощью `()`. Обратим внимание на `command.lower().split(" ")[0]`, эта цепочка методов приводит введенную команду к нижнему регистру, разделяет строку по пробелу на элементы списка и берёт самый первый элемент. Таким образом наша программа становится не чувствительной к регистру и игнорирует случайно

введенные символы через пробел. Выполнение команды представлено на рисунке 1.

Введите команду: СпИсОк

№	Название	Созвездие	Путь от Солнца (в св.г.)	Масса (в мас. Сол.)
1	Сириус	Вольшой Пёс	9	2
2	Альнирак	Орион	817	4
3	Антарес	Скорпион	550	13
4	Проксима Центавра	Альфа Центавра	4	7
5	Шторм	Анкара	159	78

Введите команду:

Рисунок 1 – Ввод команды с буквами разного регистра

Как можно заметить часть процедуры заключена в конструкцию `try:` ... `except KeyError:` ... . Она необходима на тот случай, когда человек введет команду которая содержит синтаксические ошибки или не существует вообще. В нашей процедуре мы создали словарь, в который внесли ключи – названия команд и значения – названия процедуры. При обращении к словарю по неправильному ключу (несуществующей команде) интерпретатор Python выдаст ошибку `KeyError`, на эту ошибку и реагирует наша конструкция `try: ... except ...:` выдавая сообщение о ошибке в запросе. Работа конструкции представлена на рисунке 2.

Введите команду: помоги

Такой команды не существует!

Введите команду:

Рисунок 2 – Ввод команды, которая не может быть обработана

### 1.3. Команды «помощь» и «выход»

Листинг 3 – Процедура вывода списка команд и закрытия приложения

```
def close_console():
    """Закрытие программы"""
    print("Программа закрыта!")
```

```

exit(1)

def help_commands():
    """Выдача всех команд"""
    print("«Каталог звезд нашей галактики»\n\nСписок возможных команд:\n"
          "добавить - добавление звезды\n"
          "удалить - удаление выбранной звезды\n"
          "список - вывод всех звезд в виде таблицы\n"
          "сортировать - сортировка звезд по выбранному параметру\n"
          "фильтр - вывод всех звезд, у которых масса не превосходит заданной величины\n"
          "перевод - вывод расстояния от Солнца до звезд в парсеках\n"
          "помощь - вывод возможных команд\nвыход - выход из программы\n")

```

В листинге 3 приведены две процедуры. `help_commands()`, которая выводит список существующих команд, на тот случай если надо проверить правильность написания команд или вспомнить их. И `close_console()`, которая запускает процесс закрытия программы. Выполнение команд представлено на рисунке 3.

```

Введите команду: помощь

Список возможных команд:
добавить - добавление звезды
удалить - удаление выбранной звезды
список - вывод всех звезд в виде таблицы
сортировать - сортировка звезд по выбранному параметру
фильтр - вывод всех звезд, у которых масса не превосходит заданной величины
перевод - вывод расстояния от Солнца до звезд в парсеках
помощь - вывод возможных команд
выход - выход из программы

Введите команду: выход

Программа закрыта!
|

```

Рисунок 3 – Ввод команды «помощь» и выход из программы

## 1.4. Команда «добавить»

### Листинг 4 – Добавление нового объекта

```

star_dict = {"Сириус": ("Большой Пёс", 9, 2),
             "Поллукс": ("Близнецы", 34, 2),
             "Альнитак": ("Орион", 817, 4),
             "Антарес": ("Скорпион", 550, 13),
             "Проксима Центавра": ("Альфа Центавра", 4, 7)}

```



```

def add_obj():
    while True:
        if len(star_dict) < 12:
            while True:
                name_star = input("Введите название звезды: ")
                if name_star not in [i * " " for i in range(10)]:
                    break
                else:
                    print("Название не может быть пустым!")
            while True:
                name_constellation = input("Введите название созвездия: ")
                if name_constellation not in [i * " " for i in range(10)]:
                    break
                else:
                    print("Название не может быть пустым!")
            while True:
                try:
                    distance = int(input("Введите расстояние в световых годах
(округлите до целого числа): "))
                    break
                except ValueError:
                    print("Данные не являются целым числом. Попробуйте ещё
раз.")
            while True:
                try:
                    weight = int(input("Введите массу в массах Солнца (округлите
до целого числа): "))
                    break
                except ValueError:
                    print("Данные не являются целым числом. Попробуйте ещё
раз.")
            print(f"Будет создан новый объект:\nЗвезда:
{name_star}\nСозвездие: {name_constellation}\n"
                  f"Расстояние от Солнца (в св. годах): {distance}\nМасса (в
массах Солнца): {weight}")
            while True:
                confirmation = input("Верны ли введенные данные? (д/н)")
                if confirmation == "д":
                    star_dict[name_star] = (name_constellation, distance,
weight)
                    print(f"Новый объект добавлен!")
                    break
                elif confirmation != ["д", "н"]:
                    print(f"Неизвестный ответ. Попробуйте ещё раз.")
            break
        else:
            print("Невозможно добавить новый объект!\nУдалите один объект,
чтобы добавить новый.")
            break

```

Пользователь должен иметь возможность добавлять новые объекты в каталог приложения. Реализация этого процесса представлена в листинге 4. Процедура `add_obj()` представляет собой несколько вечных циклов, заключенных в ещё один цикл. Каждый цикл ожидает от пользователя

некоторую информацию о добавляемом объекте (название звезды, название созвездия, расстояние и масса). При корректном вводе данных, осуществляется выход из одного и запуск следующего цикла. Последний цикл выводит введенные данные на проверку и ожидает подтверждения. Для хранения объектов каталога, в глобальной области создан словарь звезд, ключом является название самой звезды, а значением – множество данных об этой звезде. Перед началом запроса данных о новом объекте, проходит проверка на количество имеющихся звезд. Максимально возможное число объектов – 12. Исполнение команды представлено на рисунке 4.

```
Введите команду: добавить
Введите название звезды: Северная
Введите название созвездия: Большая медведица
Введите расстояние в световых годах (округлите до целого числа): 123
Введите массу в массах Солнца (округлите до целого числа): 3

Будет создан новый объект:
Звезда: Северная
Созвездие: Большая медведица
Расстояние от Солнца (в св. годах): 123
Масса (в массах Солнца): 3
Верны ли введенные данные? (д/н)д

Новый объект добавлен!

Введите команду: добавить

Невозможно добавить новый объект!
Удалите один объект, чтобы добавить новый.

Введите команду:
```

Рисунок 4 – Выполнение команды «добавить», попытка добавить 13-й объект

## 1.5. Команда «удалить»

### Листинг 5 – Удаление объекта

```
def del_obj():
    key = list(star_dict.keys())
    print("Звезды которые можно удалить: ")
    print(f"--|{key[0]}|", end="--")
    for i in range(1, len(key)-1):
        print(f"|{key[i]}|", end="--")
    print(f"|{key[-1]}|", end="--\n")
    while True:
        deleted = input("Выберите какую звезду будем удалять: ")
```

```

try:
    del star_dict[deleted]
    print("Звезда успешно удалена")
    break
except KeyError:
    print("Такой звезды не существует. Попробуйте ещё раз")

```

Также в программе имеется возможность удалить какую-либо имеющуюся звезду. Реализация процедуры представлена в листинге 5.

Функция `del_obj()` выводит список ключей (названий звезд), и запрашивает один ключ для удаления объекта. Исполнение команды представлено на рисунке 5.

```

Введите команду: удалить
Звезды которые можно удалить:
--|Сириус|--|Поллукс|--|Альнитак|--|Антарес|--|Проксима Центавра|--

Выберите какую звезду будем удалять (ввести название учитывая регистр): Альнитак
Звезда успешно удалена

Введите команду:

```

Рисунок 5 – Выполнение команды «удалить»

## 1.6. Команда «список»

### Листинг 6 – Вывод всех объектов в виде таблицы

```

try:
    from prettytable import PrettyTable
except ModuleNotFoundError:
    import subprocess
    import sys
    package = 'PrettyTable'
    print(f"Внимание! Будет установлен недостающий модуль\n{package}\n")
    subprocess.check_call([sys.executable, '-m', 'pip', 'install',
package])

th = ["№", "Название", "Созвездие", "Расстояние от Солнца (в св.г.)",
"Масса (в массах Солнца)"]

def list_obj(sort_key: list = None):
    columns = len(th)
    table = PrettyTable(th)
    td_data = []
    number = 1
    if sort_key is None:
        key = star_dict
    else:

```

```

    key = sort_key
    for i in key:
        td_data.append(number)
        td_data.append(i)
        td_data.append(star_dict[i][0])
        td_data.append(star_dict[i][1])
        td_data.append(star_dict[i][2])
        number += 1
    while td_data:
        table.add_row(td_data[:columns])
        td_data = td_data[columns:]
    print(table)

```

Следующая команда которая может быть выполнена приложением – вывод всех объектов в виде таблицы (листинг 6).

В данной ситуации два варианта реализации. Или продумывать систему контроля отступов в выводимой таблице, для правильного и визуально удобного чтения таблицы, или воспользоваться существующими библиотеками. В листинге 1.6 была использована библиотека PrettyTable [5]. Чтобы мы могли воспользоваться данной библиотекой, её надо импортировать. Воспользуемся конструкцией `try: ... except ModuleNotFoundError:` на тот случай, если данный модуль не был установлен ранее (в случае возникновения ошибки, нужная библиотека установится перед запуском приложения, заранее предупредив пользователя).

В глобальной области переменных создаем список из названий колонок (в последствии он пригодится и в других процедурах). В самой процедуре подсчитываем количество колонок, создаем таблицу на основе списка колонок. Преобразуем словарь `star_dict` в список `td_data`. И построчно будем добавлять данные из списка в созданную таблицу. Завершающим шагом выводим таблицу пользователю. Выполнение команды представлено на рисунке 6.

Введите команду: список				
№	Название	Созвездие	Путь от Солнца (в св.г.)	Масса (в мас. Сол.)
1	Сириус	Большой Пёс	9	2
2	Поллукс	Близнецы	34	2
3	Альнирак	Орион	817	4
4	Антарес	Скорпион	550	13
5	Проксима Центавра	Альфа Центавра	4	7
Введите команду:				

Рисунок 6 – Выполнение команды «список»

## 1.7. Команда «сортировать»

### Листинг 7 – Выбор параметров сортировки

```
def sort_obj():
    while True:
        try:
            par_sort = input("Выберите параметр сортировки
                             (название/созвездие/расстояние/масса): ")
            dict_sort = {"название": name_sort,
                         "созвездие": constellation_sort,
                         "расстояние": distance_sort, "масса": weight_sort}
            dict_sort[par_sort.split(" ")[0]]()
            break
        except KeyError:
            print("Нет такого параметра!")
```

Функция `sort_obj()` (листинг 7) по своему методу работы повторяет функцию `executing_the_command()` (листинг 2), тот же принцип создания ссылки на функцию, реализованный через обращение к словарю.

## 1.8. Команда «сортировать»

### Листинг 8 – Сортировка по выбранному параметру

```
def sorting_selection():
    """Выбор сортировки"""
    while True:
        par_sort = input("Сортировать по возрастанию или убыванию? (+/-): ")
        if par_sort in ["+", "-"]:
            break
        else:
            print("Неизвестный параметр! Укажите \"+" - возрастание или \"-\" - убывание.")
            if par_sort == "+":
                par_sort = False
            else:
                par_sort = True
    return par_sort
```

```

def weight_sort():
    """Сортировка списка по массе"""
    par_sort = sorting_selection
    keys = list(star_dict.keys())
    part_star_dict = {}
    for i in keys:
        part_star_dict[i] = star_dict[i][2]
    sorted_dict = {}
    sorted_keys = sorted(part_star_dict, key=part_star_dict.get,
reverse=par_sort())
    for w in sorted_keys:
        sorted_dict[w] = part_star_dict[w]
    list_obj(list(sorted_dict.keys()))

def distance_sort():
    """Сортировка списка по расстоянию"""
    par_sort = sorting_selection
    keys = list(star_dict.keys())
    part_star_dict = {}
    for i in keys:
        part_star_dict[i] = star_dict[i][1]
    sorted_dict = {}
    sorted_keys = sorted(part_star_dict, key=part_star_dict.get,
reverse=par_sort())
    for w in sorted_keys:
        sorted_dict[w] = part_star_dict[w]
    list_obj(list(sorted_dict.keys()))

def constellation_sort():
    """Сортировка списка по созвездию"""
    par_sort = sorting_selection
    keys = list(star_dict.keys())
    part_star_dict = {}
    for i in keys:
        part_star_dict[i] = star_dict[i][0]
    sorted_dict = {}
    sorted_keys = sorted(part_star_dict, key=part_star_dict.get,
reverse=par_sort())
    for w in sorted_keys:
        sorted_dict[w] = part_star_dict[w]
    list_obj(list(sorted_dict.keys()))

def name_sort():
    name = list(star_dict.keys())
    par_sort = sorting_selection
    name.sort(reverse=par_sort())
    list_obj(name)

```

В листинге 8 представлено 5 функций. Процедуры `name_sort()`, `weight_sort()`, `distance_sort()`, `constellation_sort()` имеют схожую структуру. `name_sort()` отличается своей простотой, т.к. сортировка осуществляется по названиям ключей. Остальные 3 процедуры

сложнее. В каждой из них создается новый временный словарь который хранит общий для всех словарей ключ (название объекта) и значение по которому проходит сортировка. Далее ключи сортируются по значению. Программа может сортировать значения по убыванию или возрастанию. Для этого вызывается функция `sorting_selection()`, которая уточняет этот параметр у человека и возвращает `True/False`.

На основе отсортированных ключей создается итоговый словарь, который подобен глобальному `star_dict`, отличием является порядок данных.

При выполнении этих процедур нам также необходимо выводить таблицу. Для оптимизации программы мы будем вызывать процедуру `list_obj()` (листинг 6), и передавать ей новый словарь данных. Пример выполнения функций представлен на рисунке 7.

Введите команду: сортировать				
Выберите параметр сортировки (название/созвездие/расстояние/масса): название				
Сортировать по возрастанию или убыванию? (+/-): -				
№	Название	Созвездие	Путь от Солнца (в св.г.)	Масса (в мас. Сол.)
1	Сириус	Большой Пёс	9	2
2	Проксима Центавра	Альфа Центавра	4	7
3	Поллукс	Близнецы	34	2
4	Антарес	Скорпион	550	13
5	Альнитак	Орион	817	4
Введите команду: сортировать				
Выберите параметр сортировки (название/созвездие/расстояние/масса): созвездие				
Сортировать по возрастанию или убыванию? (+/-): +				
№	Название	Созвездие	Путь от Солнца (в св.г.)	Масса (в мас. Сол.)
1	Проксима Центавра	Альфа Центавра	4	7
2	Поллукс	Близнецы	34	2
3	Сириус	Большой Пёс	9	2
4	Альнитак	Орион	817	4
5	Антарес	Скорпион	550	13
Введите команду:				

Рисунок 7 – Выполнение команды «сортировать»

## 1.9. Команда «перевод»

### Листинг 1.9 – Изменение данных одной графы и вывод списка

```
def convert_obj():
    """Вывод расстояния до всех звезд в парсеках вместо световых
    лет"""
    th1 = ["№", "Название", "Созвездие", "Расстояние от Солнца (в
    парсеках)", "Масса (в массах Солнца)"]
    columns = len(th1)
    table = PrettyTable(th1)
    td_data = []
    number = 1
    for i in star_dict:
        td_data.append(number)
        td_data.append(i)
        td_data.append(star_dict[i][0])
        td_data.append(round(star_dict[i][1] * 0.306, 2))
        td_data.append(star_dict[i][2])
        number += 1
    while td_data:
        table.add_row(td_data[:columns])
        td_data = td_data[columns:]
    print(table)
```

Одна из возможностей программы, это вывод таблицы данных, но с изменением единиц измерения расстояния от Солнца до объекта. Принцип работы идентичен работе процедуры `list_obj()` (листинг 6). Единственное отличие, в момент создания списка, значение расстояния умножается на 0,306. Это коэффициент различия одного светового года к парсеку. Итог работы функции представлен на рисунке 8.

Введите команду: перевод

№	Название	Созвездие	Путь от Солнца (в парсеках)	Масса (в мас. Сол.)
1	Сириус	Вольшой Пёс	2.75	2
2	Поллукс	Влизнецы	10.4	2
3	Альнитак	Орион	250.0	4
4	Антарес	Скорпион	168.3	13
5	Проксима Центавра	Альфа Центавра	1.22	7

Введите команду:

Рисунок 8 – Выполнение команды «перевод»

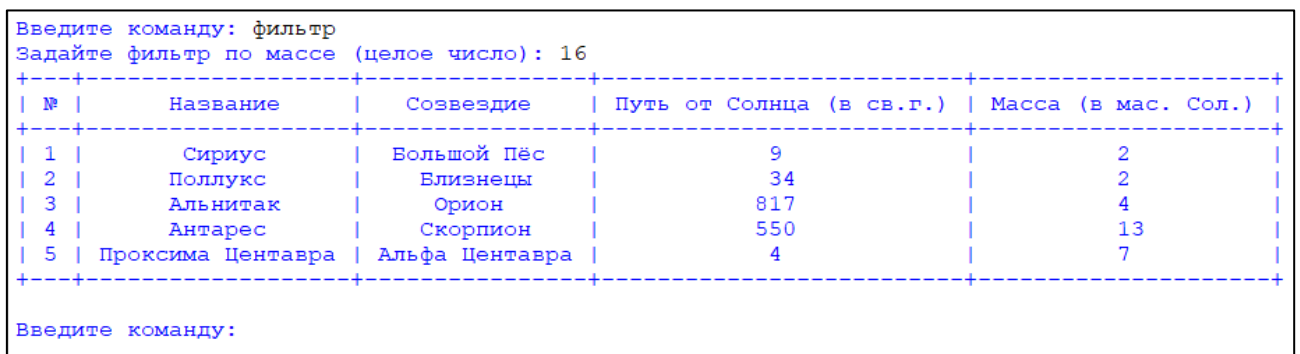


## 1.10. Команда «фильтр»

### Листинг 10 – Вывод объектов подходящих по условие

```
def filter_obj():
    """Вывод всех звезд, у которых масса не превосходит заданной
    величины"""
    columns = len(th)
    table = PrettyTable(th)
    td_data = []
    number = 1
    while True:
        try:
            weight_user = int(input("Задайте фильтр по массе (целое число):
"))
            break
        except ValueError:
            print("Данные не являются целым числом. Попробуйте ещё раз.")
    for i in star_dict:
        if star_dict[i][2] <= weight_user:
            td_data.append(number)
            td_data.append(i)
            td_data.append(star_dict[i][0])
            td_data.append(star_dict[i][1])
            td_data.append(star_dict[i][2])
            number += 1
    while td_data:
        table.add_row(td_data[:columns])
        td_data = td_data[columns:]
    print(table)
```

Последняя функция консольного приложения, фильтрация имеющихся данных (листинг 10). Суть работы схожа с принципом работы функции `list_obj()` (листинг 6). Для работы дополнительно запрашивается параметр для фильтрации, и запись в список данных происходит, когда заданный параметр больше или равен значению в у объекта в каталоге.



```
Введите команду: фильтр
Задайте фильтр по массе (целое число): 16
```

№	Название	Созвездие	Путь от Солнца (в св.г.)	Масса (в мас. Сол.)
1	Сириус	Большой Пёс	9	2
2	Поллукс	Близнецы	34	2
3	Альнитак	Орион	817	4
4	Антарес	Скорпион	550	13
5	Проксима Центавра	Альфа Центавра	4	7

```
Введите команду:
```

Рисунок 9 - Выполнение команды «фильтр»

## 2. СПИСОК ТЕСТОВЫХ СЛУЧАЕВ ДЛЯ РУЧНОГО ТЕСТИРОВАНИЯ ПРОГРАММЫ

Написанную программу следует проверить на правильность выполнения команд и общую функциональность. В приведенной таблице 1, перечислены возможные действия пользователя и ожидаемый результат от выполнения этих действий.

Таблица 1 – Тест-план для ручного тестирования программы

№	Описание тестового случая (выполняемые действия)	Ожидаемый результат
1	Запуск приложения	При первом запуске возможно сообщение «Внимание! Будет установлен недостающий модуль «PrettyTable»», Вывод названия программы и списка существующих команд: ««Каталог звезд нашей галактики» Список возможных команд: добавить - добавление звезды удалить - удаление выбранной звезды список - вывод всех звезд в виде таблицы сортировать - сортировка звезд по выбранному параметру фильтр - вывод всех звезд, у которых масса не превосходит заданной величины перевод - вывод расстояния от Солнца до звезд в парсеках помощь - вывод возможных команд выход - выход из программы»
2	Ввод команды «Добавить»	Поочередный запрос данных об объекте (название, наз. созвездия, расстояние от солнца (в св.г.), масса объекта (в массах Солнца)). Вывод введенных данных на проверку. Сообщение «Невозможно добавить новый объект! Удалите один объект, чтобы добавить новый.» в случае, если была попытка добавить 13-й объект
3	Ввод команды «Удалить»	Вывод возможных объектов для удаления. После ввода названия объекта, вывод сообщения «Звезда успешно удалена»
4	Ввод команды «Сортировать»	Запрос параметров сортировки (название объекта/название созвездия/расстояние/масса; для каждого параметра по убыванию или возрастанию). Вывод списка отсортированного по выбранному параметру (таблица)
5	Ввод команды «Фильтр»	Запрос параметра фильтрации. Вывод всех звезд, у которых масса не превосходит заданной величины

Продолжение таблицы 1

6	Ввод команды «Перевод»	Вывод списка объектов с изменённой колонкой «Расстояние от Солнца» (таблица). Вместо световых лет, расстояние в парсеках
7	Ввод команды «Список»	Вывод всех объектов в каталоге (таблица)
8	Ввод команды «Помощь»	Вывод сообщения: «Список возможных команд: добавить - добавление звезды удалить - удаление выбранной звезды список - вывод всех звезд в виде таблицы сортировать - сортировка звезд по выбранному параметру фильтр - вывод всех звезд, у которых масса не превосходит заданной величины перевод - вывод расстояния от Солнца до звезд в парсеках помощь - вывод возможных команд выход - выход из программы»
9	Ввод команды «Выход»	Сообщение «Программа закрыта!». Выход из программы (закрытие окна)
10	Ввод некорректной или не существующей команды (например, «добавить», «отфильтруй», «1-9авм*тв3»)	Сообщение «Такой команды не существует!»

### 3. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕАТУРЫ

1. Python 3.\*.\* documentation. — Текст: электронный // python.org: [сайт]. — URL: <https://docs.python.org/3/index.html> (дата обращения: 05.04.2023).
2. Консоль Windows. — Текст: электронный // Microsoft.com: [сайт]. — URL: <https://learn.microsoft.com/ru-ru/windows/console/consoles> (дата обращения: 05.04.2023).
3. Руководство пользователя Терминала. — Текст: электронный // Apple.com: [сайт]. — URL: <https://support.apple.com/ru-ru/guide/terminal/welcome/mac> (дата обращения: 05.04.2023).
4. xterm - terminal emulator for X. — Текст: электронный // ubuntu.com: [сайт]. — URL: <https://manpages.ubuntu.com/manpages/bionic/en/man1/xterm.1.html> (дата обращения: 05.04.2023).
5. Описание проекта PrettyTable. — Текст: электронный // pypi: [сайт]. — URL: <https://pypi.org/project/prettytable/> (дата обращения: 05.04.2023).