

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития  
Кафедра информационных систем и технологий**

Отчет по лабораторной работе №2,6

Дисциплина: «Основы программной инженерии»

**Выполнил:**

Студент группы ПИЖ-б-о-22-1,  
направление подготовки: 09.03.04

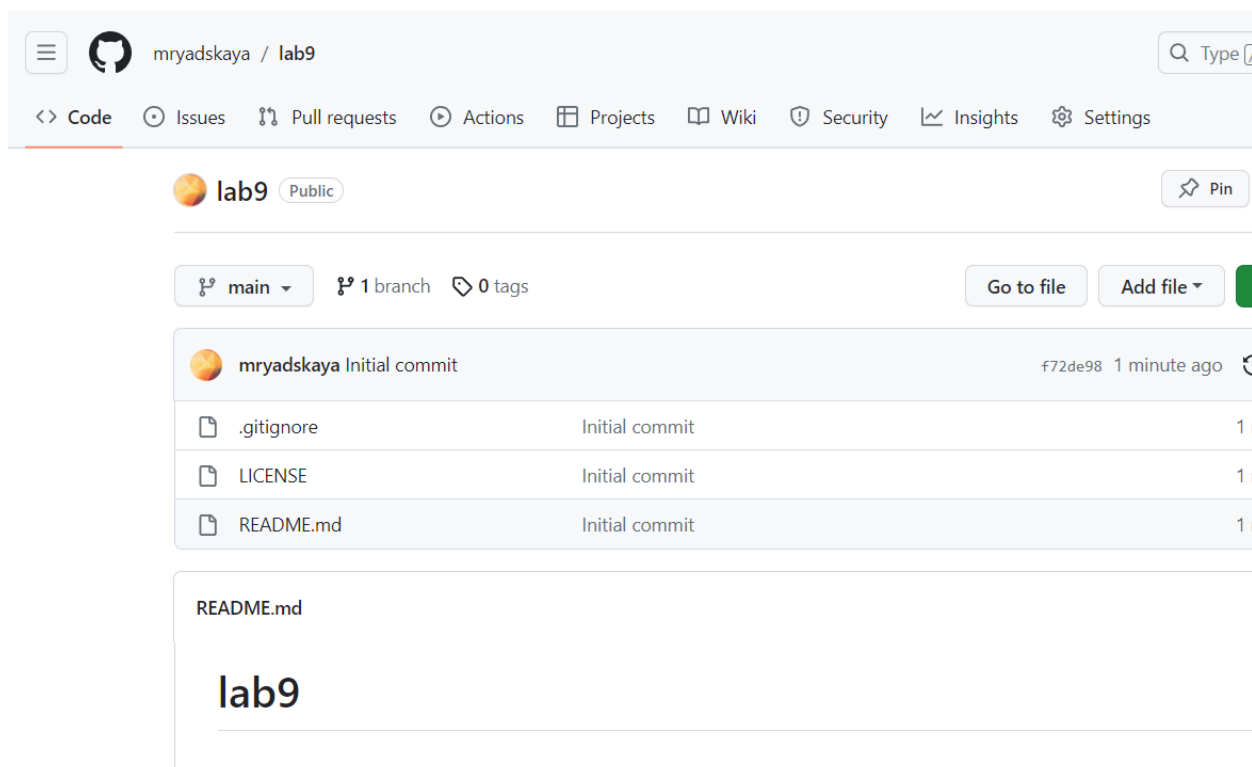
«Программная инженерия»

ФИО: Рядская Мария Александровна

**Проверил:**

Воронкин Р. А.

Ставрополь 2023



- Изучила теоретический материал работы.

Создала репозиторий на git.hub.

- Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\git1>git clone https://github.com/mryadskaya/lab9.git
Cloning into 'lab9'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\git1>cd lab9

C:\git1\lab9>git branch develop

C:\git1\lab9>git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/mryadskaya/lab9/pull/new/develop
remote:
To https://github.com/mryadskaya/lab9.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\git1\lab9>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\git1\lab9>
```

Проработать примеры лабораторной работы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    # Список работников.
    workers = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.

        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.
            name = input("Фамилия и инициалы? ")
            post = input("Должность? ")
            year = int(input("Год поступления? "))

            # Создать словарь.
            worker = {'name': name, 'post': post, 'year': year}

            # Добавить словарь в список.
            workers.append(worker)
```

```
# Отсортировать список в случае необходимости.  
if len(workers) > 1:  
    workers.sort(key=lambda item: item.get('name', ''))
```

```
elif command == 'list':  
    # Заголовок таблицы.  
    line = '+--{}-+-{}-+-{}-+-{}-+'.format(  
        *args: '-' * 4,  
        '-' * 30,  
        '-' * 20,  
        '-' * 8  
    )  
    print(line)
```

```
# Вывести данные о всех сотрудниках.  
for idx, worker in enumerate(workers, 1):  
    print(  
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(  
            *args: idx,  
            worker.get('name', ''),  
            worker.get('post', ''),  
            worker.get('year', 0)  
        )  
    )  
    print(line)
```

```
elif command.startswith('select '):
```

```
# Получить текущую дату
```

```

        parts = command.split(sep: ' ', maxsplit=1)

        # Получить требуемый стаж.
        period = int(parts[1])

        # Инициализировать счетчик.
        count = 0

        # Проверить сведения работников из списка.
        for worker in workers:
            if today.year - worker.get('year', today.year) >= period:
                count += 1
        print(
            '{:>4}: {}'.format(*args: count, worker.get('name', ''))
        )

        # Если счетчик равен 0, то работники не найдены.
        if count == 0:
            print("Работники с заданным стажем не найдены.")

    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить работника;")
        print("list - вывести список работников;")
        print("select <стаж> - запросить работников со стажем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")

    else:

```

he\_ == '\_\_main\_\_' > while True > elif command.startswith('select...

```

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\пример.py
>>> add
Фамилия и инициалы? Рядская МА
Должность? студент
Год поступления? 2022
>>> list
+-----+-----+-----+
| 1 | Рядская МА | студент | 2022 |
+-----+-----+-----+
>>>

```

## Задание 1

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      school = {'1a': 25, '1b': 27, '2b': 30, '6a': 24, '7v': 28,}
8      school['1a'] = 30
9      school['3c'] = 26
10     del school['2b']
11
12     # Находим сумму
13     total_students = sum(school.values())
14
15     # Вывод полученных значений
16     print(school)
17     print(f"Общее количество учащихся в школе: {total_students}")
```

if \_\_name\_\_ == '\_\_main\_\_'

Run задание 1 ×

↻ ■ ⋮

↑  
↓  
⇌  
⇓  
📄

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab4\PyCharm\задание 1.py"

{'1a': 30, '1b': 27, '6a': 24, '7v': 28, '3c': 26}

Общее количество учащихся в школе: 135

Process finished with exit code 0

## Задание2

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      m = {1: "A", 2: "B", 3: "C", 4: "D"}
8      dict_items = {v: k for k, v in m.items()}
9
10     print(dict_items)
```

if \_\_name\_\_ == '\_\_main\_\_'

Run задание 2 ×

↻ ■ ⋮

↑  
↓  
⇌  
⇓  
📄

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe "C:\git1\lab4\PyCharm\задание 2.py"

{'A': 1, 'B': 2, 'C': 3, 'D': 4}

Process finished with exit code 0

15:Использовать словарь, содержащий следующие ключи: фамилия, имя; знак Зодиака; дата

рождения (список из трех чисел). Написать программу, выполняющую следующие действия:

ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи

должны быть упорядочены по датам рождения; вывод на экран информации о людях,

родившихся под знаком, название которого введено с клавиатуры; если таких нет, выдать на

дисплей соответствующее сообщение.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    print("Список команд:\n")
    print("add - добавить информацию;")
    print("list - вывести список ;")
    print("select <тип> - вывод на экран фамилия, имя; знак Зодиака; дата рождения ")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

    # Список работников.
    birthday = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        match command:
            case 'exit':
                break
```



```

# Запросить данные о работнике.
name = input("Фамилия и имя? ")
zodiac = input("Знак зодиака? ")
date = input("Дата рождения? ")

# Создать словарь.
i = {'name': name, 'zodiac': zodiac, 'data': date}

# Добавить словарь в список.
birthday.append(i)

# Отсортировать список в случае необходимости.
if len(birthday) > 1:
    birthday.sort(key=lambda item: item.get('data', ''))

```

```

case 'list':
    # Заголовок таблицы.
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        *args: '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)

# Вывести данные о всех сотрудниках.
for idx, i in enumerate(birthday, 1):

```

```

print(
    '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
        *args: idx,
        i.get('name', ''),
        i.get('zodiac', ''),
        i.get('data', '')
    )
)
print(line)

case 'select ':
    # Разбить команду на части для выделения номера года.
    parts = input("Введите значение: ")

    # Проверить сведения работников из списка.
    count = 0
    for i in birthday:
        for k, v in i.items():
            if v == parts:
                print("Фамилия и имя - ", i["name"])
                print("Знак зодиака - ", i["zodiac"])
                count += 1

    # Если счетчик равен 0, то работники не найдены.
    if count == 0:
        print("Информация не найдена.")

```

```
>>> list
+-----+-----+-----+
| 1 | светлокова соня | рак | 17.07.2004 |
| 2 | ryadsks maria | рак | 17.07.2005 |
| 3 | скичко алина | рыбы | 25.02.2005 |
+-----+-----+-----+

>>> add
Фамилия и имя? ryad m
Знак зодиака? rak
дата рождения? 17.07.2005
>>> add
Фамилия и имя? svetl sonya
Знак зодиака? rak
дата рождения? 17.07.2004
>>> add
Фамилия и имя? skichko alina
Знак зодиака? fish
дата рождения? 25.02.2003
>>> list
+-----+-----+-----+
| 1 | светлокова соня | рак | 17.07.2004 |
| 2 | svetl sonya | рак | 17.07.2004 |
| 3 | ryadsks maria | рак | 17.07.2005 |
| 4 | ryad m | рак | 17.07.2005 |
| 5 | skichko alina | fish | 25.02.2003 |
| 6 | скичко алина | рыбы | 25.02.2005 |
+-----+-----+-----+

>>>
```

- Зафиксировал все изменения в github в ветке develop.

Слила ветки

```

C:\git1\lab9>git add .
C:\git1\lab9>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.
py"
        new file:   "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.
py"
        new file:   "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\32
1\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
        new file:   "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab9>git commit -m"сщхранение"
[develop c5ecc6f] сщхранение
4 files changed, 213 insertions(+)
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 1.
py"
 create mode 100644 "PyCharm/\320\267\320\260\320\264\320\260\320\275\320\270\320\265 2.
py"
 create mode 100644 "PyCharm/\320\270\320\275\320\264\320\270\320\262\320\270\320\264\32
1\203\320\260\320\273\321\214\320\275\320\276\320\265.py"
 create mode 100644 "PyCharm/\320\277\321\200\320\270\320\274\320\265\321\200.py"

C:\git1\lab9>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 3.04 KiB | 779.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/mryadskaya/lab9.git
 f72de98..c5ecc6f  develop -> develop

C:\git1\lab9>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git1\lab9>git merge develop
Updating f72de98..c5ecc6f
Fast-forward
 ...0\320\264\320\260\320\275\320\270\320\265 1.py" | 17 ++++
 ...0\320\264\320\260\320\275\320\270\320\265 2.py" | 10 +++

```

Контрольные вопросы:

1. Что такое словари в языке Python?

Словарь ( dict ) представляет собой структуру данных (которая ещё называется ассоциативный массив), предназначенную для хранения произвольных объектов с доступом по ключу. Данные в словаре хранятся в

формате ключ – значение.

2. Может ли функция `len()` быть использована при работе со словарями?

Да, функция `len()` может быть использована для работы со словарями в

Python. Она возвращает количество элементов (пар ключ-значение) в словаре.

3. Какие методы обхода словарей Вам известны?

- Цикл `for` по ключам
- Использование метода `items()`, который возвращает пары ключзначение
- Обход только ключей с использованием метода `keys()`
- Обход только значений с использованием метода `values()`

4. Какими способами можно получить значения из словаря по ключу?

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
```

```
for value in my_dict.values():
```

```
    print(value)
```

5. Какими способами можно установить значение в словаре по ключу?

```
my_dict = { }
```

```
my_dict['ключ'] = 'значение'
```

6. Что такое словарь включений?

Словарь включений аналогичен списковым включениям, за исключением

того, что он создаёт объект словаря вместо списка.

```
{x: x * x for x in (1, 2, 3, 4)}
```

```
{1: 1, 2: 4, 3: 9, 4: 16}
```

7. Самостоятельно изучите возможности функции `zip()` приведите примеры ее использования.

Функция `zip()` в Python используется для объединения двух или более итерируемых объектов (списков, кортежей, и т. д.) в один объект, создавая пары

значений. Это может быть полезно, когда вам нужно объединить данные из

нескольких источников. Вот примеры использования функции `zip()`.

```
names = ['Анна', 'Петр', 'Мария']
```

```
scores = [85, 92, 78]
```

```
student_data = list(zip(names, scores))
```

```
print(student_data)
```

Результат:

```
[('Анна', 85), ('Петр', 92), ('Мария', 78)]
```

8. Самостоятельно изучите возможности модуля `datetime`. Каким функционалом по работе с датой и временем обладает этот модуль?

Модуль `datetime` в Python предоставляет обширный функционал для работы

с датой и временем. Вот некоторые из его основных возможностей

## 1.Создание объектов даты и времени:

- `datetime.date`: Представляет дату (год, месяц, день).
- `datetime.time`: Представляет время (час, минута, секунда, микросекунда).
- `datetime.datetime`: Представляет комбинацию даты и времени.

## 2.Получение текущей даты и времени:

- `datetime.datetime.now()`: Возвращает текущую дату и время.

## 3.Разбор и форматирование даты и времени:

- `datetime.datetime.strptime()`: Разбор строки в объект `datetime`.
- `datetime.datetime.strftime()`: Преобразование объекта `datetime` в строку с заданным форматом.

## 4.Арифметика с датой и временем:

- Можно выполнять операции сложения и вычитания времени и даты, а также вычислять разницу между двумя моментами времени.

## 5.Работа с таймзонами:

- Модуль `datetime` поддерживает работу с часовыми поясами и таймзонами.

## 6.Извлечение информации:

- Можно получать год, месяц, день, часы, минуты, секунды и другую информацию о дате и времени.

## 7.Выполнение сравнений:

- Можно сравнивать даты и времена на предмет того, какой из них раньше или позже.

## 8.Работа с интервалами времени:

- Модуль `datetime` поддерживает интервалы времени, которые позволяют выражать продолжительность времени.