**Билет 8**

1. Конструкторы классов. Простые конструкторы. Конструктор копий. Вызов конструктора базового класса.

Конструктор класса — это специальный метод, который вызывается при создании объекта этого класса. В Python для этого используется метод \_\_init\_\_. Рассмотрим несколько типов конструкторов:

**Простые конструкторы**

Простой конструктор — это конструктор, который инициализирует объект с заданными значениями или по умолчанию.

class MyClass**:**

def \_\_init\_\_**(**self**,** value**):**

self**.**value **=** value # Инициализация атрибута 'value'

# Пример создания объекта

obj **=** MyClass**(**10**)**

print**(**obj**.**value**)** # Выведет 10

#### Конструктор копий

Конструктор копий используется для создания нового объекта, который является копией уже существующего. В Python это может быть сделано с помощью копирования объекта.

import copy

class MyClass**:**

def \_\_init\_\_**(**self**,** value**):**

self**.**value **=** value

def \_\_copy\_\_**(**self**):**

**return** MyClass**(**self**.**value**)** # Возвращаем новый объект с теми же значениями

# Пример создания объекта и его копии

obj1 **=** MyClass**(**10**)**

obj2 **=** copy**.**copy**(**obj1**)**

print**(**obj2**.**value**)** # Выведет 10

#### Вызов конструктора базового класса

Если класс наследуется от другого, то мы можем вызвать конструктор базового класса с помощью функции super().

class BaseClass**:**

def \_\_init\_\_**(**self**,** base\_value**):**

self**.**base\_value **=** base\_value

class DerivedClass**(**BaseClass**):**

def \_\_init\_\_**(**self**,** base\_value**,** derived\_value**):**

super**().**\_\_init\_\_**(**base\_value**)** # Вызов конструктора базового класса

self**.**derived\_value **=** derived\_value

# Пример создания объекта производного класса

obj **=** DerivedClass**(**5**,** 10**)**

print**(**obj**.**base\_value**)** # Выведет 5

print**(**obj**.**derived\_value**)** # Выведет 10

2. Используя объектно-ориентированный подход (допускается STL) написать программу, выполняющую следующие действия:   
а) Создание динамического массива строк   
б) Реализовать сортировку массива по числу вхождений слов из массива-словаря.

#### а) Создание динамического массива строк

Для создания динамического массива строк можем использовать стандартный список в Python.

class StringArray**:**

def \_\_init\_\_**(**self**):**

self**.**arr **=** **[]** # Инициализируем пустой список

def add**(**self**,** string**):**

self**.**arr**.**append**(**string**)** # Добавляем строку в массив

# Пример создания массива строк и добавления элементов

string\_array **=** StringArray**()**

string\_array**.**add**(**"hello"**)**

string\_array**.**add**(**"world"**)**

print**(**string\_array**.**arr**)** # Выведет **[**'hello'**,** 'world'**]**

#### б) Реализовать сортировку массива по числу вхождений слов из массива-словаря

Для сортировки массива по числу вхождений слов из словаря, можно использовать метод сортировки списка с использованием ключа. В качестве ключа будет использоваться количество вхождений каждого слова из словаря в строки массива.

class StringArray**:**

def \_\_init\_\_**(**self**,** words\_dict**):**

self**.**arr **=** **[]** # Массив строк

self**.**words\_dict **=** words\_dict # Словарь**,** где ключи **-** слова**,** а значения **-** их веса или важность

def add**(**self**,** string**):**

self**.**arr**.**append**(**string**)** # Добавляем строку в массив

def sort\_by\_word\_frequency**(**self**):**

# Сортируем строки по числу вхождений слов из словаря

self**.**arr**.**sort**(**key**=**self**.**\_count\_word\_frequency**,** reverse**=**True**)**

def \_count\_word\_frequency**(**self**,** string**):**

# Считаем количество вхождений слов из словаря в строку

count **=** 0

**for** word in self**.**words\_dict**:**

count **+=** string**.**count**(**word**)** # Считаем вхождения каждого слова из словаря в строку

**return** count

# Пример создания массива строк и словаря для сортировки

words\_dict **=** **{**"hello"**:** 1**,** "world"**:** 2**}**

string\_array **=** StringArray**(**words\_dict**)**

string\_array**.**add**(**"hello world"**)**

string\_array**.**add**(**"hello there"**)**

string\_array**.**add**(**"goodbye world"**)**

# Сортируем и выводим результат

string\_array**.**sort\_by\_word\_frequency**()**

print**(**string\_array**.**arr**)** # Выведет строки**,** отсортированные по числу вхождений слов из словаря

В этом примере программа создает динамический массив строк и сортирует его по количеству вхождений слов из заданного словаря. Метод sort\_by\_word\_frequency использует вспомогательный метод \_count\_word\_frequency, который вычисляет, сколько раз слова из словаря встречаются в строке.