**Билет 9**

1. Деструкторы классов: простые, виртуальные. Привести примеры.

**Деструктор** — это специальный метод класса, который вызывается, когда объект этого класса уничтожается. В нем обычно очищаются ресурсы, которые были выделены для объекта (например, освобождение памяти).

**Простой деструктор** — это обычный деструктор, который просто освобождает ресурсы объекта. Он вызывается автоматически при удалении объекта.

**Пример простого деструктора**:

#include <iostream>

**using** **namespace** std**;**

class MyClass **{**

public**:**

MyClass**()** **{**

cout **<<** "Конструктор вызван!" **<<** endl**;**

**}**

**~**MyClass**()** **{**

cout **<<** "Деструктор вызван!" **<<** endl**;**

**}**

**};**

int main**()** **{**

MyClass obj**;** // Создается объект, вызывается конструктор

// Деструктор вызовется автоматически, когда объект выйдет из области видимости

**return** 0**;**

**}**

**Виртуальный деструктор** — это деструктор, который позволяет правильно уничтожать объекты производных классов через указатель на базовый класс. Это необходимо для того, чтобы при удалении объекта через указатель на базовый класс, сначала вызывался деструктор производного класса, а затем базового.

**Пример виртуального деструктора**:

#include <iostream>

**using** **namespace** std**;**

class Base **{**

public**:**

virtual **~**Base**()** **{**

cout **<<** "Деструктор базового класса!" **<<** endl**;**

**}**

**};**

class Derived **:** public Base **{**

public**:**

**~**Derived**()** override **{**

cout **<<** "Деструктор производного класса!" **<<** endl**;**

**}**

**};**

int main**()** **{**

Base**\*** obj **=** **new** Derived**();** // Создаем объект производного класса

**delete** obj**;** // Деструктор будет вызван в правильном порядке

**return** 0**;**

**}**

2. Используя объектно-ориентированный подход (допускается STL) написать программу, выполняющую с использованием ООП следующие действия:   
а) Создание динамического массива символов:   
б) Шифровка и дешифровка строки методом перестановок (ключ для перестановок произвольной длины).

Для этого создадим класс, который будет работать с динамическим массивом символов и реализовывать метод шифровки и дешифровки строки методом перестановок с произвольным ключом.  
#include <iostream>

#include <vector>

#include <string>

**using** **namespace** std**;**

class Cipher **{**

private**:**

vector**<**char**>** arr**;** // Динамический массив символов

public**:**

// Конструктор для создания динамического массива

Cipher**(**string str**)** **{**

**for** **(**char c **:** str**)** **{**

arr**.**push\_back**(**c**);**

**}**

**}**

// Функция для шифровки строки методом перестановок

string encrypt**(**string key**)** **{**

string encrypted **=** arr**.**data**();**

int n **=** encrypted**.**size**();**

int keyLength **=** key**.**size**();**

// Перестановка символов на основе ключа

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

encrypted**[**i**]** **=** arr**[(**i **+** key**[**i **%** keyLength**]** **-** '0'**)** **%** n**];** // сдвиг по ключу

**}**

**return** encrypted**;**

**}**

// Функция для дешифровки строки методом перестановок

string decrypt**(**string encrypted**,** string key**)** **{**

string decrypted **=** encrypted**;**

int n **=** decrypted**.**size**();**

int keyLength **=** key**.**size**();**

// Восстановление строки, делая обратную перестановку

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

decrypted**[**i**]** **=** arr**[(**i **-** **(**key**[**i **%** keyLength**]** **-** '0'**)** **+** n**)** **%** n**];** // обратный сдвиг

**}**

**return** decrypted**;**

**}**

**};**

int main**()** **{**

string str **=** "HelloWorld"**;**

string key **=** "314"**;** // Ключ для перестановок (случайный порядок)

Cipher cipher**(**str**);**

// Шифруем строку

string encrypted **=** cipher**.**encrypt**(**key**);**

cout **<<** "Зашифрованная строка: " **<<** encrypted **<<** endl**;**

// Дешифруем строку

string decrypted **=** cipher**.**decrypt**(**encrypted**,** key**);**

cout **<<** "Расшифрованная строка: " **<<** decrypted **<<** endl**;**

**return** 0**;**

**}**

 **Динамический массив символов** создается с помощью vector<char>.

 **Шифровка и дешифровка** — перестановка символов в строке на основе ключа. Мы выполняем сдвиг символов в зависимости от значений ключа, который представляет собой строку, где каждый символ — это цифра (например, "314").