МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчет по лабораторной №1

IP – Адресация

по дисциплине «Протоколы компьютерного взаимодействия»

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил | студент гр. ФИБ-4301-51-00 |  | Кочкин В.Р. |
| Проверил | преподаватель каф. ПМиИ |  | Белиц А.Б. |

Киров 2022

Цель работы

Познакомиться с ip-адресацией и действиями над ip и маской подсети.

Задания

**Задание 1**

В ходе лабораторной работы необходимо разработать программу, которая по IP-адресу и маске выведет: IP-адрес сети, IP-адрес узла, максимальное количество узлов в сети, IP-адрес для широковещательной рассылки.

**Задание 2**

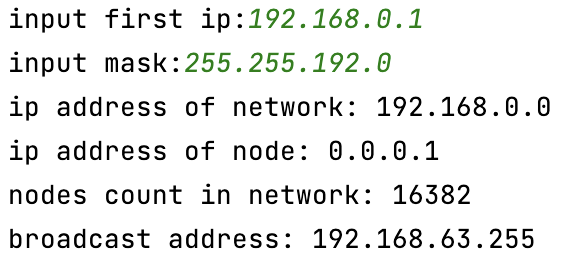
В ходе лабораторной работы необходимо разработать программу, которая определяет, относятся ли два узла к одной подсети (даны IP-адреса двух компьютеров и маска подсети).

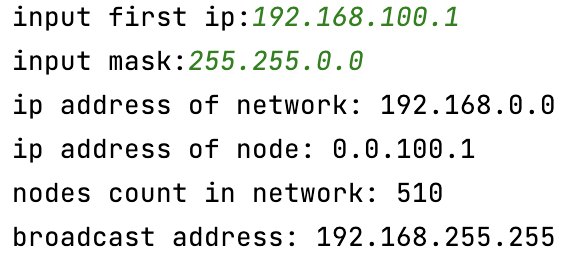
**Задание 3**

В ходе лабораторной работы необходимо разработать программу, которая определяет минимальную маску подсети если известны IP-адреса нескольких узлов из этой подсети.

Полученные результаты

**Задание 1**



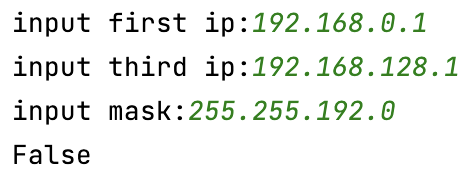


Листинг программы приведен в [приложении А задание 1](#_Задание_1.).

**Задание 2**

Изображение выглядит как текст

Автоматически созданное описание



Листинг программы приведен в [приложении А Задание 2](#_Задание_2.).

**Задание 3**

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Листинг программы приведен в [приложении А Задание 3](#_Задание_3.).

Вывод

В ходе лабораторной работы я познакомился с ip-адресацией и действиями над ip и маской подсети, которые были реализованы в виде программы на языке python.

# **Приложения**

## **Приложение А. Листинги программ**

Константы и вспомогательные функции.

const.py

IP\_LENGTH = 4  
BYTE\_MAX\_VALUE = 255  
BIT\_IN\_BYTE = 8

helper.py

def ip\_to\_str(ip: []) -> str:  
 return '.'.join(str(byte) for byte in ip)

### Задание 1.

import const  
import helper  
  
  
def get\_network\_address(ip, mask) -> list:  
 network\_address = []  
 for i in range(const.IP\_LENGTH):  
 network\_address.append(ip[i] & mask[i])  
 return network\_address  
  
  
def get\_node\_address(ip, mask) -> list:  
 node\_address = []  
 for i in range(const.IP\_LENGTH):  
 node\_address.append(ip[i] & (const.BYTE\_MAX\_VALUE - mask[i]))  
 return node\_address  
  
  
def get\_count\_nodes(mask) -> int:  
 degree = 0  
 i = 0  
  
 while mask[i] == const.BYTE\_MAX\_VALUE and i < const.BYTE\_MAX\_VALUE:  
 i += 1  
  
 degree += bin(mask[i]).count("0") - 1  
  
 while i != const.IP\_LENGTH - 1:  
 degree += const.BIT\_IN\_BYTE  
 i += 1  
  
 return 2 \*\* degree - 2  
  
  
def get\_broadcast\_address(network\_address, mask) -> list:  
 broadcast\_address = network\_address  
 inverted\_mask = get\_invert\_ip(mask)  
 for i in range(const.IP\_LENGTH):  
 if mask[i] != const.BYTE\_MAX\_VALUE:  
 broadcast\_address[i] = network\_address[i] + inverted\_mask[i]  
  
 return broadcast\_address  
  
  
def get\_invert\_ip(ip: []) -> list:  
 result = []  
 for i in range(const.IP\_LENGTH):  
 result.append(const.BYTE\_MAX\_VALUE - ip[i])  
 return result  
  
  
def print\_ip\_info(ip: [], mask: []) -> None:  
 network\_address = get\_network\_address(ip, mask)  
 print("ip address of network:", helper.ip\_to\_str(network\_address))  
  
 node\_address = get\_node\_address(ip, mask)  
 print("ip address of node:", helper.ip\_to\_str(node\_address))  
  
 count\_nodes = get\_count\_nodes(mask)  
 print("nodes count in network:", count\_nodes)  
  
 broadcast\_address = get\_broadcast\_address(network\_address, mask)  
 print("broadcast address:", helper.ip\_to\_str(broadcast\_address))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ip = list(map(int, input("input first ip:").split(".")))  
 mask = list(map(int, input("input mask:").split(".")))print\_ip\_info(ip, mask)

### Задание 2.

import task\_1  
  
  
def is\_equal\_subnet(first\_ip, third\_id, mask) -> bool:  
 return task\_1.get\_network\_address(first\_ip, mask) == task\_1.get\_network\_address(third\_id, mask)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 first\_ip = list(map(int, input("input first ip:").split(".")))  
 third\_ip = list(map(int, input("input third ip:").split(".")))  
 mask = list(map(int, input("input mask:").split(".")))  
print(is\_equal\_subnet(first\_ip, third\_ip, mask))

### Задание 3.

import const  
import helper  
  
  
def get\_minimal\_subnet\_mask(first\_ip: [], third\_ip: []) -> list:  
 mask = []  
 count\_equal\_octets = get\_count\_equal\_octets(first\_ip, third\_ip)  
 for i in range(count\_equal\_octets):  
 mask.append(const.BYTE\_MAX\_VALUE)  
  
 mask.append(calculate\_mask\_byte(first\_ip, third\_ip, count\_equal\_octets))  
  
 while len(mask) != const.IP\_LENGTH:  
 mask.append(0)  
  
 return maskdef get\_count\_equal\_octets(first\_ip: [], third\_ip: []) -> int:  
 result = 0  
 i = 0  
 while first\_ip[i] == third\_ip[i]:  
 result += 1  
 i += 1  
 return result  
  
  
def calculate\_mask\_byte(first\_ip: [], third\_ip: [], byte: int) -> int:  
 pointer = 128  
 mask = 0  
  
 while pointer & first\_ip[byte] == pointer & third\_ip[byte]:  
 mask += pointer  
 pointer >>= 1  
  
 return mask  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 first\_ip = list(map(int, input("input first ip:").split(".")))  
 third\_ip = list(map(int, input("input third ip:").split(".")))  
print(helper.ip\_to\_str(get\_minimal\_subnet\_mask(first\_ip, third\_ip)))