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

```
input first ip:192.168.0.1
input mask:255.255.192.0
ip address of network: 192.168.0.0
ip address of node: 0.0.0.1
nodes count in network: 16382
broadcast address: 192.168.63.255
```

```
input first ip:192.168.100.1
input mask:255.255.0.0
ip address of network: 192.168.0.0
ip address of node: 0.0.100.1
nodes count in network: 510
broadcast address: 192.168.255.255
```

Листинг программы приведен в приложении А задание 1.

Задание 2

```
input first ip:192.168.1.0
input third ip:192.168.0.1
input mask:255.255.0.0
True

input first ip:192.168.0.1
input third ip:192.168.128.1
input mask:255.255.192.0
False
```

Листинг программы приведен в приложении А Задание 2.

Задание 3

```
input first ip:127.0.0.1
input third ip:127.2.0.1
255.252.0.0

input first ip:192.168.1.0
input third ip:192.168.10.0
255.255.240.0
```

Листинг программы приведен в приложении А 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 mask


def 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)))
```