МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент: *Попов Матвей Романович, группа М8О-208Б-20*
Преподаватель: *Дорохов Евгений Павлович*

Задание

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод. Реализовать над объектами реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Вариант 18

Создать класс IPAddress для работы с адресом в интернет. Класс состоит из четырех чисел unsigned char (a,b,c,d). Реализовать арифметические операции сложения, вычитания, а также операции сравнения (для сравнение на больше/меньше считать что левые байты главнее т.е. вначале сравниваются первые байты, потом вторые и т.д.). Так же реализовать функцию, которая будет определять принадлежность адреса к подсети по адресу подсети (a1,b1,c1,d1) и битовой маске подсети (a2,b2,c2,d2). Например, адрес 192.168.1.30 принадлежит подсети 192.168.0.0 с маской 255.255.0.0.

Описание программы

Программа состоит из 3 файлов: main.cpp, IP.cpp и IP.h, содержит класс IPAdress, конструктор по умолчанию, перегрузки операторов сравнения, сложения, вычитания и пользовательский литерал, инкапсуляция соблюдена.

Дневник отладки

При отладке ошибок в выполнении программы не выявлено.

Выводы

Проделав лабораторную работу, познакомился с системой пользовательских литералов.

Листинг

main.cpp

```
#include <iostream>
#include <stdio.h>
#include <string>
#include "IP.h"

std::string operator "" _with_dots(const char* s, size_t size)
{
    std::string str;
    for (int i = 0; i < 16; ++i)
    {
        if (s[i] == ' ')
```

```
        {
            str.push_back('.');
        }
        else
        {
            str.push_back(s[i]);
        }
    }
    return str;
}

int main()
{
    std::cout << "Enter A and B IPAddresses:\n";
    int a, b, c, d;
    std::cin >> a >> b >> c >> d;
    IPAddress A(a, b, c, d);
    std::cin >> a >> b >> c >> d;
    IPAddress B(a, b, c, d);
    if (A == B)
    {
        std::cout << "IPAdresses are equal\n";
    }
    if (A > B)
    {
        std::cout << "A is greater than B\n";
    }
    if (A < B)
    {
        std::cout << "B is greater than A\n";
    }
    std::cout << "Sum of A and B is ";
    (A + B).Print();
    std::cout << "Difference of A and B is ";
    (A - B).Print();
    std::cout << "Enter the mask: \n";
    std::cin >> a >> b >> c >> d;
    IPAddress M(a, b, c, d);
    if (A.Check(B, M))
    {
        std::cout << "A belongs to B\n";
    }
    else
    {
        std::cout << "A do not belongs to B" << std::endl;
    }
    std::cout << "Using of the literal: \n";
    //char* s = M.Get();
    std::cout << "192 168 10 12"_with_dots << std::endl;
    return 0;
}
```

IP.h

```
#ifndef IP_H
#define IP_H

class IPAddress
{
public:
    IPAddress();
    IPAddress(unsigned char _a, unsigned char _b, unsigned char _c, unsigned char _d);
```

```
    friend IPAddress operator+(IPAddress A, IPAddress B);
    friend IPAddress operator-(IPAddress A, IPAddress B);
    friend bool operator==(IPAddress A, IPAddress B);
    friend bool operator!=(IPAddress A, IPAddress B);
    friend bool operator<(IPAddress A, IPAddress B);
    friend bool operator>(IPAddress A, IPAddress B);
    friend bool operator<=(IPAddress A, IPAddress B);
    friend bool operator>=(IPAddress A, IPAddress B);

    void Print();

    bool Check(IPAddress Addr, IPAddress Mask);

private:
    unsigned char a, b, c, d;
};

#endif
```

IP.cpp

```
#include "IP.h"
#include <iostream>
#include <stdio.h>

IPAddress::IPAddress() : a(0), b(0), c(0), d(0)
    {}

IPAddress::IPAddress(unsigned char _a, unsigned char _b, unsigned char _c, unsigned char _d) :
a(_a), b(_b), c(_c), d(_d)
    {}

void IPAddress::Print()
{
    printf("%d %d %d %d\n", a, b, c, d);
}

bool IPAddress::Check(IPAddress Addr, IPAddress Mask)
{
    if (Mask.a == 0)
    {
        return ((Addr.a == 0)&&(Addr.b == 0)&&(Addr.c == 0)&&(Addr.d == 0));
    }
    if (Mask.a < 255)
    {
        return ((Addr.b == 0)&&(Addr.c == 0)&&(Addr.d == 0)&&(Mask.a + a - 255 == Addr.a));
    }
    if (Mask.b == 0)
    {
        return ((a == Addr.a)&&(Addr.b == 0)&&(Addr.c == 0)&&(Addr.d == 0));
    }
    if (Mask.b < 255)
    {
        return ((a == Addr.a)&&(Mask.b + b - 255 == Addr.b)&&(Addr.c == 0)&&(Addr.d == 0));
    }
    if (Mask.c == 0)
    {
        return ((a == Addr.a)&&(b == Addr.b)&&(Addr.c == 0)&&(Addr.d == 0));
    }
    if (Mask.c < 255)
    {
        return ((a == Addr.a)&&(b == Addr.b)&&(Mask.c + c - 255 == Addr.c)&&(Addr.d == 0));
    }
```

```
    if (Mask.d == 0)
    {
        return ((a == Addr.a)&&(b == Addr.b)&&(c == Addr.c)&&(Addr.d == 0));
    }
    if (Mask.d < 255)
    {
        return ((a == Addr.a)&&(b == Addr.b)&&(c == Addr.c)&&(Mask.d + d - 255 == Addr.d));
    }
    return true;
}

IPAddress operator+(IPAddress A, IPAddress B)
{
    unsigned _a = (A.a + B.a) % 256;
    unsigned _b = (A.b + B.b) % 256;
    unsigned _c = (A.c + B.c) % 256;
    unsigned _d = (A.d + B.d) % 256;
    return IPAddress(_a, _b, _c, _d);
}

IPAddress operator-(IPAddress A, IPAddress B)
{
    int _a = (A.a - B.a) % 256;
    int _b = (A.b - B.b) % 256;
    int _c = (A.c - B.c) % 256;
    int _d = (A.d - B.d) % 256;
    return IPAddress(_a, _b, _c, _d);
}

bool operator==(IPAddress A, IPAddress B)
{
    return ((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c)&&(A.d == B.d));
}

bool operator!=(IPAddress A, IPAddress B)
{
    return !((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c)&&(A.d == B.d));
}

bool operator>(IPAddress A, IPAddress B)
{
    if ((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c))
    {
        return A.d > B.d;
    }
    if ((A.a == B.a)&&(A.b == B.b))
    {
        return A.c > B.c;
    }
    if (A.a == B.a)
    {
        return A.b > B.b;
    }
    return A.a > B.a;
}

bool operator<(IPAddress A, IPAddress B)
{
    if ((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c))
    {
        return A.d < B.d;
    }
    if ((A.a == B.a)&&(A.b == B.b))
```

```
    {
        return A.c < B.c;
    }
    if (A.a == B.a)
    {
        return A.b < B.b;
    }
    return A.a < B.a;
}

bool operator>=(IPAddress A, IPAddress B)
{
    if ((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c))
    {
        return A.d >= B.d;
    }
    if ((A.a == B.a)&&(A.b == B.b))
    {
        return A.c > B.c;
    }
    if (A.a == B.a)
    {
        return A.b > B.b;
    }
    return A.a > B.a;
}

bool operator<=(IPAddress A, IPAddress B)
{
    if ((A.a == B.a)&&(A.b == B.b)&&(A.c == B.c))
    {
        return A.d <= B.d;
    }
    if ((A.a == B.a)&&(A.b == B.b))
    {
        return A.c < B.c;
    }
    if (A.a == B.a)
    {
        return A.b < B.b;
    }
    return A.a < B.a;
}
```