**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Епанешников Владислав Сергеевич

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

# Постановка задачи

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. в начале сравниваются первые байты, потом вторые и т.д.). Также 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. Операции сложения, вычитания и сравнения необходимо реализовать в виде перегрузки операторов. Необходимо реализовать пользовательский литерал для работы с константами типа IPAddress.

# Описание программы

I. Класс IPAddress

a) Конструктор по умолчанию и параметризированный конструктор.

б) Геттеры для приватных полей (pound, shilling, penny).

в) Функция, которая позволяет вычислить принадлежность адреса к подсети

II. а) Перегрузка операторов сложения, вычитания, умножения, деления и операторов ввода и вывода.

б) Перегрузка операторов сравнения.

III. Главная функция

а) Пользовательский строковый литерал для класса IPAddress.

# Код программы

/\*\*

\* Епанешников М80-206Б-19

\* 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. в начале сравниваются первые байты,

\* потом вторые и т.д.). Также 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.

\* Операции сложения, вычитания и сравнения необходимо реализовать в виде перегрузки операторов.

\* Необходимо реализовать пользовательский литерал для работы с константами типа IPAddress.

\*/

#include <iostream>

typedef unsigned char uc;

using namespace std;

class IPAddress {

public:

// конструктор по умолчанию

IPAddress() {

a = 0;

b = 0;

c = 0;

d = 0;

}

// параметризированный конструктор

IPAddress(const int& a, const int& b, const int& c, const int& d) {

if (a > 255 || a < 0 || b > 255 || b < 0 || c < 0 || c > 255 || d < 0 || d > 255) {

throw out\_of\_range("Invalid values (" + to\_string(a) + "." + to\_string(b) + "." + to\_string(c) + "." + to\_string(d) + ")");

}

this->a = a;

this->b = b;

this->c = c;

this->d = d;

}

// getters

uc GetA() const{

return a;

}

uc& GetA() {

return a;

}

uc GetB() const{

return b;

}

uc& GetB() {

return b;

}

uc GetC() const{

return c;

}

uc& GetC() {

return c;

}

uc GetD() const{

return d;

}

uc& GetD() {

return d;

}

private:

// байты ip адреса

uc a;

uc b;

uc c;

uc d;

};

// операторы сравнения

bool operator==(const IPAddress& lhs, const IPAddress& rhs) {

return lhs.GetA() == rhs.GetA() && lhs.GetB() == rhs.GetB() && lhs.GetC() == rhs.GetC() && lhs.GetD() == rhs.GetD();

}

bool operator!=(const IPAddress& lhs, const IPAddress& rhs) {

return !(lhs.GetA() == rhs.GetA() && lhs.GetB() == rhs.GetB() && lhs.GetC() == rhs.GetC() && lhs.GetD() == rhs.GetD());

}

bool operator<(const IPAddress& lhs, const IPAddress& rhs) {

if (lhs.GetA() != rhs.GetA()) {

return lhs.GetA() < rhs.GetA();

}

if (lhs.GetB() != rhs.GetB()) {

return lhs.GetB() < rhs.GetB();

}

if (lhs.GetC() != rhs.GetC()) {

return lhs.GetC() < rhs.GetC();

}

return lhs.GetD() < rhs.GetD();

}

bool operator>(const IPAddress& lhs, const IPAddress& rhs) {

if (lhs.GetA() != rhs.GetA()) {

return lhs.GetA() > rhs.GetA();

}

if (lhs.GetB() != rhs.GetB()) {

return lhs.GetB() > rhs.GetB();

}

if (lhs.GetC() != rhs.GetC()) {

return lhs.GetC() > rhs.GetC();

}

return lhs.GetD() > rhs.GetD();

}

// оператор сложения

IPAddress operator+(const IPAddress& lhs, const IPAddress& rhs) {

uc a = lhs.GetA() + rhs.GetA() > 255 ? 255 : lhs.GetA() + rhs.GetA();

uc b = lhs.GetB() + rhs.GetB() > 255 ? 255 : lhs.GetB() + rhs.GetB();

uc c = lhs.GetC() + rhs.GetC() > 255 ? 255 : lhs.GetC() + rhs.GetC();

uc d = lhs.GetD() + rhs.GetD() > 255 ? 255 : lhs.GetD() + rhs.GetD();

return {a, b, c, d};

}

// оператор вычитания

IPAddress operator-(const IPAddress& lhs, const IPAddress& rhs) {

uc a = lhs.GetA() - rhs.GetA() < 0 ? 0 : lhs.GetA() - rhs.GetA();

uc b = lhs.GetB() - rhs.GetB() < 0 ? 0 : lhs.GetB() - rhs.GetB();

uc c = lhs.GetC() - rhs.GetC() < 0 ? 0 : lhs.GetC() - rhs.GetC();

uc d = lhs.GetD() - rhs.GetD() < 0 ? 0 : lhs.GetD() - rhs.GetD();

return {a, b, c, d};

}

// оператор конъюнкции

IPAddress operator&(const IPAddress& lhs, const IPAddress& rhs) {

uc a = lhs.GetA() & rhs.GetA();

uc b = lhs.GetB() & rhs.GetB();

uc c = lhs.GetC() & rhs.GetC();

uc d = lhs.GetD() & rhs.GetD();

return {a, b, c, d};

}

// функция, проверяющая принадлежность адреса к подсети по адресу подсети и маски

bool CheckIp(const IPAddress& ip, const IPAddress& sub, const IPAddress& mask) {

return (ip & mask) == sub;

}

// оператор ввода

istream& operator>>(istream& stream, IPAddress& ip) {

int a, b, c, d;

stream >> a;

stream.ignore(1);

stream >> b;

stream.ignore(1);

stream >> c;

stream.ignore(1);

stream >> d;

stream.ignore(1);

ip = {a, b, c, d};

return stream;

}

// оператор вывода

ostream& operator<<(ostream& stream, const IPAddress& ip) {

stream << (int) ip.GetA() << '.' << (int) ip.GetB() << '.' << (int) ip.GetC() << '.' << (int) ip.GetD();

return stream;

}

// строковый литерал для класса IPAddress

const IPAddress operator"" \_ip(const char \*str, size\_t size) {

int val = 0;

int a[4] = {0, 0, 0, 0};

int k = 0;

for (int i = 0; i < size; ++i) {

if (str[i] == '.') {

a[k++] = val;

val = 0;

continue;

}

val = 10 \* val + str[i] - '0';

}

a[3] = val;

return {a[0], a[1], a[2], a[3]};

}

int main() {

try {

IPAddress sub = "192.168.0.0"\_ip, mask = "255.255.0.0"\_ip, a, b;

cin >> a >> b;

cout << "Operator+ : " << a << " + " << b << " = " << a + b << endl;

cout << "Operator- : " << a << " - " << b << " = " << a - b << endl;

cout << "Operator< : ";

if (a < b) {

cout << a << " < " << b << " (меньше)" << endl;

}

else {

cout << a << " >= " << b << " (не меньше)" << endl;

}

cout << "Operator> : ";

if (a > b) {

cout << a << " > " << b << " (больше)" << endl;

}

else {

cout << a << " >= " << b << " (не больше)" << endl;

}

cout << "Operator== : ";

if (a == b) {

cout << a << " = " << b << endl;

}

else {

cout << a << " != " << b << endl;

}

cout << "CheckIp: ";

if (CheckIp(a, sub, mask)) {

cout << a << " belongs to the subnet" << endl;

}

else {

cout << "No" << endl;

}

} catch(exception& ex) {

cout << ex.what() << endl;

}

return 0;

}

# Набор тестов.

Тест №1:

251.5.1.0

50.1.62.122

Результат:

MacBook:lab2 vladislove$ ./a.out < test1.txt

Operator+ : 251.5.1.0 + 50.1.62.122 = 255.6.63.122

Operator- : 251.5.1.0 - 50.1.62.122 = 201.4.0.0

Operator< : 251.5.1.0 >= 50.1.62.122 (не меньше)

Operator> : 251.5.1.0 > 50.1.62.122 (больше)

Operator== : 251.5.1.0 != 50.1.62.122

CheckIp: 251.5.1.0 doesn't belong to the subnet

Тест №2:

192.168.1.30

15.52.5.1.7

Результат:

Operator+ : 192.168.1.30 + 15.52.5.1 = 207.220.6.31

Operator- : 192.168.1.30 - 15.52.5.1 = 177.116.0.29

Operator< : 192.168.1.30 >= 15.52.5.1 (не меньше)

Operator> : 192.168.1.30 > 15.52.5.1 (больше)

Operator== : 192.168.1.30 != 15.52.5.1

CheckIp: 192.168.1.30 belongs to the subnet

Тест №3:

256.12.62.7

-1.6.3.7

Результат:

Invalid values (256.12.62.7)

# Вывод

Данная лабораторная работа помогает изучить классы и их методы в С++. Выполнив её, можно научиться перегружать операторы, а также создавать пользовательские литералы.

# Список литературы

1. Информационный ресурс CppStudio

http://cppstudio.com/post/439/.

2. Электронная энциклопедия Ravesli

https://ravesli.com/urok-116-konstruktory/.

3. Статья про перегрузку операторов

https://tproger.ru/translations/cpp-operator-overload-p1/.

4. Оператор множественного выбора

http://easy-code.ru/lesson/switch-case-cpp.