**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: Чернобаев Андрей Александрович

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата: 12.10.20

Оценка:

Москва, 2020

1. Постановка задачи

**задача N 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.  Операции сложения, вычитания и сравнения необходимо реализовать в виде перегрузки операторов.  Необходимо реализовать пользовательский литерал для работы с константами типа IPAddress. |

.

1. Описание программы

* 2 функции для реализации метода split
* класс ipAdress
  + приватные поля для a b c d
  + массив указателей на них
  + перегрузки операторов
  + метод для определение принадлежания к подсети

1. Набор тестов

из файла **test\_01.txt**

192 168 1 1 192 99 1 1 add

192 168 1 1 192 99 1 1 sub

192 168 1 1 192 99 1 1 less

192 168 1 1 192 99 1 1 greater

192 168 1 1 192 99 1 1 eq

192 168 1 30 192 168 0 0 255 255 0 0 belong\_to\_subnet

192 168 1 1 192 168 1 1 literal

из файла **test\_02.txt**

3 44 256 1 192 99 1 9 add

56 123 256 0 192 5 1 3 sub

192 168 254 245 192 99 1 5 less

192 168 14 13 78 99 1 12 greater

192 255 1 1 192 101 1 1 eq

192 167 1 30 192 168 0 0 255 255 0 0 belong\_to\_subnet

192 168 1 1 192 168 1 1 literal

1. Результаты выполнения тестов

|  |  |
| --- | --- |
| input | output |
| 192 168 1 1 192 99 1 1 add  192 168 1 1 192 99 1 1 sub  192 168 1 1 192 99 1 1 less  192 168 1 1 192 99 1 1 greater  192 168 1 1 192 99 1 1 eq  192 168 1 30 192 168 0 0 255 255 0 0 belong\_to\_subnet  192 168 1 1 192 168 1 1 literal | write exit for exit  write help for help  input format ipAdress1 ipAdress2 (ipAdress3) action  in input mode add - addition, sub - subtraction, less - ipAdress1 < ipAdress2, greater - ipAdress1 > ipAdress2, eq for ipAdress1 == ipAdress2, belong\_to\_subnet - ipAdress, subnet, mask, literal - show literal  for 192 168 1 1  out 1: 129.12.2.2  out 2: 0.69.0.0  out 3: 0  out 4: 1  out 5: 0  out 6: 1  out 7: 192.168.1.1 |
| 3 44 256 1 192 99 1 9 add  56 123 256 0 192 5 1 3 sub  192 168 254 245 192 99 1 5 less  192 168 14 13 78 99 1 12 greater  192 255 1 1 192 101 1 1 eq  192 167 1 30 192 168 0 0 255 255 0 0 belong\_to\_subnet  192 168 1 1 192 168 1 1 literal | write exit for exit  write help for help  input format ipAdress1 ipAdress2 (ipAdress3) action  in input mode add - addition, sub - subtraction, less - ipAdress1 < ipAdress2, greater - ipAdress1 > ipAdress2, eq for ipAdress1 == ipAdress2, belong\_to\_subnet - ipAdress, subnet, mask, literal - show literal  for 192 168 1 1  out 1: 195.143.1.10  out 2: 120.118.255.253  out 3: 0  out 4: 1  out 5: 0  out 6: 0  out 7: 192.168.1.1 |

1. Листинг программы

ссылка на github:

https://github.com/ruthenorum/oop\_exercise\_02

**rugewit@rugewitPC**:**/media/rugewit/WDC DATA/MAI/3 sem/OOP/my\_code/task2/clion**$ cat main.cpp

#include <iostream>

#include <climits>

#include <string>

#include <vector>

#include <sstream>

using byte = unsigned char;

template <typename Out>

void split(const std::string &s, char delim, Out result) {

std::istringstream iss(s);

std::string item;

while (std::getline(iss, item, delim)) {

\*result++ = item;

}

}

std::vector<std::string> split(const std::string &s, char delim) {

std::vector<std::string> elems;

split(s, delim, std::back\_inserter(elems));

return elems;

}

class IPAddress{

private:

byte a,b,c,d;

byte\* array[4] = {&a, &b, &c, &d};

friend std::ostream& operator<<(std::ostream& os, const IPAddress& ipAddress);

public:

IPAddress(byte \_a, byte \_b, byte \_c, byte \_d){

a = \_a;

b = \_b;

c = \_c;

d = \_d;

}

IPAddress operator+(const IPAddress& rhs) const{

byte new\_a = (1u\*a + rhs.a) % UCHAR\_MAX;

byte new\_b = (1u\*b + rhs.b) % UCHAR\_MAX;

byte new\_c = (1u\*c + rhs.c) % UCHAR\_MAX;

byte new\_d = (1u\*d + rhs.d) % UCHAR\_MAX;

return IPAddress(new\_a, new\_b, new\_c, new\_d);

}

IPAddress operator-(const IPAddress& rhs) const{

byte new\_a = (byte) (1u\*a - rhs.a);

byte new\_b = (byte) (1u\*b - rhs.b);

byte new\_c = (byte) (1u\*c - rhs.c);

byte new\_d = (byte) (1u\*d - rhs.d);

return IPAddress(new\_a, new\_b, new\_c, new\_d);

}

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

return (a == rhs.a) \* (b == rhs.b) \* (c == rhs.c) \* (d == rhs.d);

}

bool operator<(const IPAddress& rhs) const{

for(int i = 0; i < 4; i++){

if (\*(array[i]) > \*(rhs.array[i])){

return false;

}

}

return true;

}

bool operator>(const IPAddress& rhs) const{

for(int i = 0; i < 4; i++){

if (\*(array[i]) < \*(rhs.array[i])){

return false;

}

}

return true;

}

bool belong\_to\_subnet(const IPAddress& subnet, const IPAddress& mask) const{

byte new\_a = a & mask.a;

byte new\_b = b & mask.b;

byte new\_c = c & mask.c;

byte new\_d = d & mask.d;

return IPAddress(new\_a, new\_b, new\_c, new\_d) == subnet;

}

};

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

std::vector<std::string> numbers = split(str, '.');

return IPAddress(std::stoi(numbers[0]),std::stoi(numbers[1])

,std::stoi(numbers[2]),std::stoi(numbers[3]));

}

std::ostream& operator<<(std::ostream& os, const IPAddress& ipAddress){

std::string out = std::to\_string(ipAddress.a) + '.' +

std::to\_string(ipAddress.b) + '.' +

std::to\_string(ipAddress.c) + '.' +

std::to\_string(ipAddress.d);

return os << out;

}

void help(){

std::cout << "write exit for exit" << std::endl;

std::cout << "write help for help" << std::endl;

std::cout << "input format ipAdress1 ipAdress2 (ipAdress3) action" << std::endl;

std::cout << "in input mode add - addition, sub - subtraction, less - ipAdress1 < ipAdress2, "

"greater - ipAdress1 > ipAdress2, eq for ipAdress1 == ipAdress2, "

"belong\_to\_subnet - ipAdress, subnet, mask, literal - show literal for 192 168 1 1" << std::endl;

}

int main() {

std::string cmd;

std::vector<std::string> out;

std::stringstream buffer;

int count = 0;

help();

while (true){

getline(std::cin, cmd);

if (cmd == "exit"){

break;

}

else if (cmd == "help"){

help();

}

else {

count++;

byte a1,b1,c1,d1;

byte a2,b2,c2,d2;

byte a3,b3,c3,d3;

std::string action;

std::vector<std::string> x = split(cmd, ' ');

if (x.size() == 9){

a1 = stod(x[0]);

b1 = stod(x[1]);

c1 = stod(x[2]);

d1 = stod(x[3]);

a2 = stod(x[4]);

b2 = stod(x[5]);

c2 = stod(x[6]);

d2 = stod(x[7]);

action = x[8];

}

else if (x.size() == 13){

a1 = stod(x[0]);

b1 = stod(x[1]);

c1 = stod(x[2]);

d1 = stod(x[3]);

a2 = stod(x[4]);

b2 = stod(x[5]);

c2 = stod(x[6]);

d2 = stod(x[7]);

a3 = stod(x[8]);

b3 = stod(x[9]);

c3 = stod(x[10]);

d3 = stod(x[11]);

action = x[12];

}

IPAddress a = {a1, b1, c1, d1};

IPAddress b = {a2, b2, c2, d2};

IPAddress c = {a3, b3, c3, d3};

if (action == "add"){

buffer << "out " << count << ": " << (a + b) << std::endl;

}

else if (action == "sub"){

buffer << "out " << count << ": " << (a - b) << std::endl;

}

else if (action == "less"){

buffer << "out " << count << ": " << (a < b) << std::endl;

}

else if (action == "greater"){

buffer << "out " << count << ": " << (a > b) << std::endl;

}

else if (action == "eq"){

buffer << "out " << count << ": " << (a == b) << std::endl;

}

else if (action == "belong\_to\_subnet"){

buffer << "out " << count << ": " << a.belong\_to\_subnet(b, c) << std::endl;

}

else if (action == "literal"){

buffer << "out " << count << ": " << "192.168.1.1"\_ipAddr << std::endl;

}

}

}

std::cout << buffer.str() << std::endl;

return 0;

}