Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №1

по курсу «Объектно-ориентированное программирование»

III Семестр

Тема:

Простые классы

|  |  |
| --- | --- |
| Студент: | Обыденкова Ю.Ю. |
| Группа: | М80-208Б-18 |
| Преподаватель: | Журавлев А.А. |
| Вариант: | 18 |
| Оценка: |  |
| Дата: |  |

# 1. Код программы на языке С++

1. Файл IPv4Address.cpp

#include "IPAddress.h"

IPAddress::IPAddress(unsigned char p1, unsigned char p2, unsigned char p3, unsigned char p4)

: part1(p1), part2(p2), part3(p3), part4(p4) {

}

IPAddress IPAddress::Sum(const IPAddress &other) const {

unsigned char new\_part1;

unsigned char new\_part2;

unsigned char new\_part3;

unsigned char new\_part4;

if ((int)part1 + (int)other.part1 > 255) {

new\_part1 = 255;

} else {

new\_part1 = part1 + other.part1;

}

if ((int)part2 + (int)other.part2 > 255) {

new\_part2 = 255;

} else {

new\_part2 = part2 + other.part2;

}

if ((int)part3 + (int)other.part3 > 255) {

new\_part3 = 255;

} else {

new\_part3 = part3 + other.part3;

}

if ((int)part4 + (int)other.part4 > 255) {

new\_part4 = 255;

} else {

new\_part4 = part4 + other.part4;

}

return {new\_part1, new\_part2, new\_part3, new\_part4};

}

IPAddress IPAddress::Subtract(const IPAddress &other) const {

unsigned char new\_part1;

unsigned char new\_part2;

unsigned char new\_part3;

unsigned char new\_part4;

if ((int)part1 - (int)other.part1 < 0) {

new\_part1 = 0;

} else {

new\_part1 = part1 - other.part1;

}

if ((int)part2 - (int)other.part2 < 0) {

new\_part2 = 0;

} else {

new\_part2 = part2 - other.part2;

}

if ((int)part3 - (int)other.part3 < 0) {

new\_part3 = 0;

} else {

new\_part3 = part3 - other.part3;

}

if ((int)part4 - (int)other.part4 < 0) {

new\_part4 = 0;

} else {

new\_part4 = part4 - other.part4;

}

return {new\_part1, new\_part2, new\_part3, new\_part4};

}

bool IPAddress::Equal(const IPAddress &other) const {

return std::tie(part1, part2, part3, part4) == std::tie(other.part1, other.part2, other.part3, other.part4);

}

bool IPAddress::More(const IPAddress &other) const {

return std::tie(part1, part2, part3, part4) > std::tie(other.part1, other.part2, other.part3, other.part4);

}

bool IPAddress::Less(const IPAddress &other) const {

return std::tie(part1, part2, part3, part4) < std::tie(other.part1, other.part2, other.part3, other.part4);

}

bool IPAddress::BelongToSubnet(const IPAddress& subnet\_address, const IPAddress& subnet\_mask) const {

IPAddress conj = IPAddress(part1 & subnet\_mask.part1, part2 & subnet\_mask.part2, part3 & subnet\_mask.part3, part4 & subnet\_mask.part4);

return conj.Equal(subnet\_address);

}

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

return os << (int)address.part1 << "." << (int)address.part2 << "." << (int)address.part3 << "." << (int)address.part4;

}

std::istream& operator >> (std::istream& is, IPAddress& address) {

unsigned char part1;

unsigned char part2;

unsigned char part3;

unsigned char part4;

std::string str\_address;

is >> str\_address;

std::stringstream ss(str\_address);

getline(ss, str\_address,'.');

address.part1 = std::stoi(str\_address);

getline(ss, str\_address,'.');

address.part2 = std::stoi(str\_address);

getline(ss, str\_address,'.');

address.part3 = std::stoi(str\_address);

getline(ss, str\_address);

address.part4 = std::stoi(str\_address);

return is;

}

2. Файл IPv4Address.h

#pragma once

#include <iostream>

#include <tuple>

#include <string>

#include <sstream>

class IPAddress {

public:

IPAddress() = default;

IPAddress(unsigned char p1, unsigned char p2, unsigned char p3, unsigned char p4);

IPAddress Sum(const IPAddress& other) const;

IPAddress Subtract(const IPAddress& other) const;

bool More(const IPAddress& other) const;

bool Less(const IPAddress& other) const;

bool Equal(const IPAddress& other) const;

bool BelongToSubnet(const IPAddress& subnet\_address, const IPAddress& subnet\_mask) const;

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

friend std::istream& operator >> (std::istream& is, IPAddress& address);

private:

unsigned char part1 = 0;

unsigned char part2 = 0;

unsigned char part3 = 0;

unsigned char part4 = 0;

};

3. Файл main.cpp

#include <iostream>

#include "IPAddress.h"

#include <vector>

#include <string>

void print\_help() {

std::cout << "'create' - Create IpAddress" << std::endl;

std::cout << "'operation +/-' - Perform + or -" << std::endl;

std::cout << "'compare =/>/<' - Perform = or > or <" << std::endl;

std::cout << "'check\_subnet' - Check subnet" << std::endl;

std::cout << "'exit' - Exit" << std::endl;

std::cout << "'help' - Help" << std::endl;

}

int main() {

print\_help();

std::vector<IPAddress> Addresses;

std::string command;

while (std::cin >> command) {

if (command == "create") {

IPAddress new\_address;

std::cin >> new\_address;

Addresses.push\_back(new\_address);

std::cout << "Address number " << Addresses.size() << "\n"

<< "Address: " << Addresses.back() << "\n";

} else if (command == "compare") {

std::string compare\_string;

size\_t lhs, rhs;

std::cin >> compare\_string >> lhs >> rhs;

lhs--;

rhs--;

if (lhs >= Addresses.size() || rhs >= Addresses.size() || compare\_string.size() != 1

|| (compare\_string[0] != '=' && compare\_string[0] != '>' && compare\_string[0] != '<')) {

std::cout << "Incorrect parameters" << "\n";

continue;

}

char compare = compare\_string[0];

std::cout << lhs + 1 << " " << rhs + 1 << " " << compare << " ";

if (compare == '<') {

std::cout << std::boolalpha << Addresses[lhs].Less(Addresses[rhs]) << "\n";

} else if (compare == '=') {

std::cout << std::boolalpha << Addresses[lhs].Equal(Addresses[rhs]) << "\n";

} else if (compare == '>') {

std::cout << std::boolalpha << Addresses[lhs].More(Addresses[rhs]) << "\n";

}

} else if (command == "operation") {

std::string operation\_string;

int lhs, rhs;

std::cin >> operation\_string >> lhs >> rhs;

rhs--;

lhs--;

if (lhs >= Addresses.size() || rhs >= Addresses.size() || operation\_string.size() != 1

|| (operation\_string[0] != '-' && operation\_string[0] != '+')) {

std::cout << "Incorrect parameters" << "\n";

continue;

}

char operation = operation\_string[0];

std::cout << lhs + 1 << " " << operation << " " << rhs + 1 << "\n";

if (operation == '+') {

std::cout << Addresses[lhs].Sum(Addresses[rhs]) << "\n";

} else if (operation == '-') {

std::cout << Addresses[lhs].Subtract(Addresses[rhs]) << "\n";

}

} else if (command == "check\_subnet") {

std::cout << "Enter number of created address, subnet address and subnet mask" << "\n";

IPAddress subnet\_address;

IPAddress subnet\_mask;

size\_t number;

std::cin >> number >> subnet\_address >> subnet\_mask;

number--;

if (number >= Addresses.size()) {

std::cout << "Incorrect parameters" << "\n";

continue;

}

std::cout << "Address is " << Addresses[number] << "\n"

<< "Address is " << (Addresses[number].BelongToSubnet(subnet\_address, subnet\_mask) ? "" : "not ")

<< "belong to subnet " << subnet\_address << " with mask " << subnet\_mask << "\n";

}

else if (command == "help") {

print\_help ();

}

else if (command == "exit") {

break;

} else {

std::cin.ignore(32767,'\n');

std::cout << "Unknown command\n";

}

}

return 0;

}

4. Файл CmakeLists.txt

CC = g++

CPP = main.cpp IPAddress.cpp

NAME = oop\_exercise\_01

all:

$(CC) -o $(NAME) $(CPP)

clean:

rm -f \*.o $(NAME)

2. Ссылка на репозиторий на Github

<https://github.com/obydenkova/oop_exercise_01>

3.Набор testcases

<< означает входные данные, >> - выходные

1. test\_01.txt

<< create

<< 192.168.1.30

>> Address number 1

>> Address: 192.168.1.30

<< create

<< 192.168.0.0

>> Address number 2

>> Address: 192.168.0.0

<< create

<< 255.255.0.0

>> Address number 3

>> Address: 255.255.0.0

<< operation + 1 2

>> 1 + 2

>> 255.255.1.30

<< operation - 3 1

>> 3 - 1

>> 63.87.0.0

<< compare = 3 2

>> 3 2 = false

<< compare > 1 2

>> 1 2 > true

<< compare < 1 3

>> 1 3 < true

<< check\_subnet

>> Enter number of created address, subnet address and subnet mask

<< 1 2 3

>> Address is 192.168.1.30

>> Address is not belong to subnet 2.2.2.2 with mask 3.3.3.3

<< check\_subnet

>> Enter number of created address, subnet address and subnet mask

<< 1 192.168.0.0 255.255.0.0

>> Address is 192.168.1.30

>> Address is belong to subnet 192.168.0.0 with mask 255.255.0.0

<< help

>> 'create' - Create IpAddress

>> 'operation +/-' - Perform + or -

>> 'compare =/>/<' - Perform = or > or <

>> 'check\_subnet' - Check subnet

>> 'exit' - Exit

>> 'help' - Help

<< e

>> Unknown command

<< exit

2. test\_02.txt

<< operation 1 2 -

>> Incorrect parameters

<< compare 1 2 2

>> Incorrect parameters

<< create 1 2

>> Address number 1

>> Address: 1.1.1.1

>> Unknown command

<< create 43

>> Address number 2

>> Address: 43.43.43.43

<< compare > 1 2

>> 1 2 > false

<< compare + 1 2

>> Incorrect parameters

<< operation > 2 1

>> Incorrect parameters

<< operation - 2 4

>> Incorrect parameters

<< create 0.0.0.0

>> Address number 3

>> Address: 0.0.0.0

<< create -1.-1.-1.-1

>> Address number 4

>> Address: 255.255.255.255

<< operation - 3 4

>> 3 - 4

>> 0.0.0.0

<< create 5

>> Address number 5

>> Address: 5.5.5.5

<< operation + 1 2 3 4 5

>> 1 + 2

>> 44.44.44.44

>> Unknown command

<< create 5

>> Address number 6

>> Address: 5.5.5.5

<< compare = 5 6

>> 5 6 = true

4. Объяснение результатов программы

Данная программа создает класс IPv4Address для работы с IP адресами. Класс состоит из четырех октетов типа unsigned char p1, p2, p3, p4.  
  
 В программе реализованы следующие операции:

1. Сложение и вычитание

2. Сравнение на больше/меньше/равно

3. Принадлежность адреса к подсети по адресу подсети и битовой маске подсети.

Для более удобной работы с программой создано меню, которое высвечивается при открытии, а далее при надобности вызывается функцией „help“:

'create' - Create IpAddress

'operation +/-' - Perform + or -

'compare =/>/<' - Perform = or > or <

'check\_subnet' - Check subnet

'exit' - Exit

'help' - Help

create p1.p2.p3.p4 создает Ip адрес a, затем можно создать так же адрес b;

operation + a b выполняет функцию сложения с созданными адресами a и b;

operation - a b выполняет функцию вычитания с созданными адресами a и b

compare = a b сравнивает на равенство адреса a и b;

compare > a b сравнивает на знак больше адреса a и b;

compare < a b сравнивает на знак меньше адреса a и b;

check\_subnet a q.w.e.r t.y.u.i проверяется принадлежность существующего адреса a к подести по введенному адресу подести q.w.e.r и введенной битовой маске подсети t.y.u.i

help вызывает меню с доступными функциями

exit завершает работу программы

5. Вывод.

Проделав данную работу я изучила классы и научилась их применять. Сделала вывод, что класс — очень удобная структура, позволяющая представлять свойства объекта и предоставляющая удобную работу с ними. Создала класс, соответствующий варианту моего задания, реализовала для него арифметические операции сложения, вычитания, а также операции сравнения.