Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Объектно-ориентированное программирование»
III Семестр

Тема:
Операторы, литералы

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

Москва
2019

Задание:
Изучение механизмов перегрузки операторов;
Изучение механизмов работы с пользовательскими литералами;

1. Код программы на языке C++

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::operator=(const IPAddress &other) const {
    return std::tie(part1, part2, part3, part4) == std::tie(other.part1, other.part2, other.part3,
other.part4);
}

bool IPAddress::operator>(const IPAddress &other) const {
    return std::tie(part1, part2, part3, part4) > std::tie(other.part1, other.part2, other.part3,
other.part4);
}

bool IPAddress::operator<(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.operator=(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 operator>(const IPAddress& other) const;
    bool operator<(const IPAddress& other) const;
    bool operator=(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] < Addresses[rhs]) << "\n";
        } else if (compare == '=') {
            std::cout << std::boolalpha << (Addresses[lhs] = Addresses[rhs]) << "\n";
        } else if (compare == '>') {
            std::cout << std::boolalpha << (Addresses[lhs] > 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_02

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.

Благодаря перегрузке логических операторов программист, который будет использовать данный класс, получает более гибкое и простое управление над экземплярами класса, а благодаря литералам пользователь в одной строке может задать параметры экземпляра и начать работу с ним.