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

ЛАБОРАТОРНАЯ РАБОТА №2

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Примаченко Александр Александрович, группа М8О-208Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Задание:

Разработать программу на языке C++ согласно варианту задания. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод. 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:

Исходный код разделён на 3 файла:

- `IPAddress.h` – описание основных функций класса IPAddress
- `IPAddress.cpp` – реализация функционала класса IPAddress
- `main.cpp` – основная программа

Дневник отладки:

Программа в отладке не нуждалась, весь необходимый функционал был реализован без всяких заминок.

Вывод:

При выполнении лабораторной работы я познакомился на практике с одной из разновидностей специального полиморфизма, а именно с перегрузкой, что даёт возможность объявлять функции с одним и тем же названием, но с разными типами аргументов и их количеством, ну или другими словами, у функции появляется возможность иметь несколько сигнатур, что на практике бывает очень удобным. Так же познакомился с весьма полезным средством, как пользовательский литерал, который служит для улучшения читаемости кода, но, с другой стороны, может и понизить её, если использовать его не в тех местах. Как результат работы была написана программа с использованием перегрузки операторов и реализацией пользовательского литерала, что дало мне понять какие есть преимущества и недостатки в их использовании.

Исходный код:

APAddress.h:

```
#ifndef OOP_IPADRESS_H
#define OOP_IPADRESS_H

#include <iostream>
using namespace std;

typedef unsigned char uc;

class IPAddress {
public:
   IPAddress();
   IPAddress(uc _a, uc _b, uc _c, uc _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() const;

   bool Check(IPAddress Address, IPAddress Mask) const;

   char* Get() const;

private:
   uc a;
   uc b;
   uc c;
   uc d;
};


#endif //OOP_IPADRESS_H
```

APAddress.cpp:

```
#include "IPAddress.h"
#include <string>

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

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

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

bool IPAddress::Check(IPAddress Address, IPAddress Mask) const
{
   if (Mask.a == 0)
   {
      return ((Address.a == 0)&&(Address.b == 0)&&(Address.c == 0)&&(Address.d == 0));
   }
```

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

char* IPAddress::Get() const
{
    int _a = a;
    int _b = b;
    int _c = c;
    int _d = d;
    char* s = new char [16];
    int t = 0;
    string buff = to_string(_a);
    for (int i = 0; i < buff.length(); ++i)
    {
        s[t + i] = buff[i];
    }
    t += buff.length();
    s[t] = ' ';
    ++t;
    buff = to_string(_b);
    for (int i = 0; i < buff.length(); ++i)
    {
        s[t + i] = buff[i];
    }
    t += buff.length();
    s[t] = ' ';
    ++t;
    buff = to_string(_c);
    for (int i = 0; i < buff.length(); ++i)
    {
        s[t + i] = buff[i];
    }
    t += buff.length();
    s[t] = ' ';
    ++t;
    buff = to_string(_d);
    for (int i = 0; i < buff.length(); ++i)
    {
```

```
        s[t + i] = buff[i];
    }
    return s;
}

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;
}
```

Main.cpp:

```
#include "IPAddress.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 IP-Addresses:\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 << "IP-Addresses 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 << "172 22 10 10"_with_dots << std::endl;
    return 0;
}
```

Пример работы:

```
C:\Users\SashaPaladin\CLionProjects\OOP\lab0.2\cmake-build-debug\lab0.2.exe
Enter A and B IP-Addresses:
172 22 34 70
172 22 30 40
A is greater than B
Sum of A and B is 88 44 64 110
Difference of A and B is 0 0 4 30
Enter the mask:
255 255 0 0
A do not belongs to B
Using of the literal:
172.22.10.10
Process finished with exit code 0
```