**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Федоров Антон Сергеевич

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

**Вариант 19**. **Создать класс Address** для работы с адресами домов. Адрес должен состоять из строк с названием города и улицы и чисел с номером дома и квартиры. Реализовать операции сравнения адресов, а также операции проверки принадлежности адреса к улице и городу. В операциях не должен учитываться регистр строки. Так же необходимо сделать операцию, которая возвращает истину если два адреса находятся по соседству (на одной улице в одном городе и дома стоят подряд).

Операцию сравнения равенства реализовать в виде перегрузки оператора. Операцию нахождения «по соседству» реализовать в виде перегрузки оператора &.

Необходимо реализовать пользовательский литерал для работы с константами типа **Address**.

1. **Описание программы**

Класс Address имеет 4 приватные переменные: CityName(std::string), StreetName(std::string), HouseNumber(unsigned int) и ApartmentNumber(unsigned int). Имеется 2 конструктора, один строит объект класса Address по четырем явно указанным аргументам, другой принимает только один аргумент std::string и парсит его как в формате “city\_name,streen\_name,house\_name,apartment\_name ”. Публичный метод print печатает поля объекта в вышеуказанном формате. Перегруженный оператор << позволяет, аналогично print, печатать объект, но с использованием std::cout. Методы IsOnThisCity и IsOnThisStreet проверяют принадлежность адреса данному городу/улице. Так как в двух разных городах могут быть улицы с одинаковыми именами, в метод IsOnThisStreet принимает и улицу, и город. Регистр строк игнорируется, для этого используется приватный метод IgnoreRegister. Перегруженные операторы <, <=, >, >=, ==, != сравнивают два адреса в лексикографическом порядке. Перегруженный оператор & проверяет на равенство город, улицу и модуль разности номеров домов двух адресов. Если он равен единице, то оператор вернет истину, в иных случаях(даже когда адресы указывают на один и тот же дом вернет ложь). Литерал \_address принимает строку и ее размер и аналогично конструктору с одним странным аргументом парсит в объект класса Address.

1. Набор тестов

**Тест №1 Тестирование проверки корректности ввода.**

asdfds,dsf

fesfg

**Тест №2 Тестирование проверки корректности ввода.**

saf,fas,56,4

fesfg

**Тест №3 Сравнение одинаковых адресов, для отношения соседства программа должна вывести ложь.**

Korolov,Pushkinskaya,8,10

Korolov,Pushkinskaya,8,10

Moscow

Kalanchevskaya

**Тест №4 Сравнение двух соседних адресов.**

Korolov,Pushkinskaya,7,10

Korolov,Pushkinskaya,8,10

Korolov

Mayakovskogo

**Тест №5 Проверка принадлежности адресов конкретному городу и улице. Адреса с разными городами, но одинаковой улицей должны дать разные результаты.**

Korolov,Pushkinskaya,7,10

Moscow,Pushkinskaya,8,10

Korolov

Pushkinskaya

1. Результаты выполнения тестов

**Тест №1**

test literal address:Here's city name,Here's street name,12,34

input address in "city,street,house,apartment" format

insert address1:asdfds,dsf

invalid format

Process finished with exit code 0

**Тест №2**

test literal address:Here's city name,Here's street name,12,34

input address in "city,street,house,apartment" format

insert address1:saf,fas,56,4

insert address2:saf,fas,56,-4

invalid format

Process finished with exit code 0

**Тест №3**

test literal address:Here's city name,Here's street name,12,34

input address in "city,street,house,apartment" format

insert address1:Korolov,Pushkinskaya,8,10

insert address2:Korolov,Pushkinskaya,8,10

address1 is Korolov,Pushkinskaya,8,10

address2 is Korolov,Pushkinskaya,8,10

insert city name for location of address1 and address2:Moscow

is address1 located in this city? NO

is address2 located in this city? NO

insert street name for location of address1 and address2:Kalanchevskaya

is address1 located on this street? NO

is address2 located on this street NO

is address1 and address2 are neighbors? NO

is address1 == address2? YES

is address1 != address2? NO

is address1 < address2? NO

is address1 <= address2? YES

is address1 > address2? NO

is address1 >= address2? YES

Process finished with exit code 0

**Тест №4**

test literal address:Here's city name,Here's street name,12,34

input address in "city,street,house,apartment" format

insert address1:Korolov,Pushkinskaya,7,10

insert address2:Korolov,Pushkinskaya,8,10

address1 is Korolov,Pushkinskaya,7,10

address2 is Korolov,Pushkinskaya,8,10

insert city name for location of address1 and address2:Korolov

is address1 located in this city? YES

is address2 located in this city? YES

insert street name for location of address1 and address2:Mayakovskogo

is address1 located on this street? NO

is address2 located on this street NO

is address1 and address2 are neighbors? YES

is address1 == address2? NO

is address1 != address2? YES

is address1 < address2? YES

is address1 <= address2? YES

is address1 > address2? NO

is address1 >= address2? NO

Process finished with exit code 0

**Тест №5**

test literal address:Here's city name,Here's street name,12,34

input address in "city,street,house,apartment" format

insert address1:Korolov,Pushkinskaya,7,10

insert address2:Moscow,Pushkinskaya,8,10

address1 is Korolov,Pushkinskaya,7,10

address2 is Moscow,Pushkinskaya,8,10

insert city name for location of address1 and address2:Korolov

is address1 located in this city? YES

is address2 located in this city? NO

insert street name for location of address1 and address2:Pushkinskaya

is address1 located on this street? YES

is address2 located on this street? NO

is address1 and address2 are neighbors? NO

is address1 == address2? NO

is address1 != address2? YES

is address1 < address2? YES

is address1 <= address2? YES

is address1 > address2? NO

is address1 >= address2? NO

Process finished with exit code 0

1. **Листинг программы**

Класс Address вынесен в отдельный файл. Реализация методов класса вынесена в отдельный cpp файл.

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.17)

project(oop\_exercise\_02)

set(CMAKE\_CXX\_STANDARD 20)

add\_executable(oop\_exercise\_02 main.cpp Address.cpp Address.h)

**Address.h**

#ifndef OOP\_EXERCISE\_02\_ADDRESS\_H

#define OOP\_EXERCISE\_02\_ADDRESS\_H

#include <string>

#include <algorithm>

#include <cctype>

#include <iostream>

class Address{

private:

std::string CityName = "";

std::string StreetName = "";

unsigned int HouseNumber = 0;

unsigned int ApartmentNumber = 0;

std::string IgnoreRegister(const std::string str);

public:

Address(const std::string &city, const std::string &street, const unsigned int &houseNum, const unsigned int &apartmentNum);

Address(const std::string &str);

void print();

friend std::ostream & operator << (std::ostream &out, const Address &address);

void SetAddress(const std::string newCity, const std::string newStreet, const unsigned int newHouseNumber, const unsigned int newApartmentNumber);

bool IsOnThisCity(std::string City);

bool IsOnThisStreet(std::string City ,std::string Street);

friend bool operator&(const Address lhs, const Address rhs);

friend bool operator<(const Address lhs, const Address rhs);

friend bool operator<=(const Address lhs, const Address rhs);

friend bool operator>(const Address lhs, const Address rhs);

friend bool operator>=(const Address lhs, const Address rhs);

friend bool operator==(const Address lhs, const Address rhs);

friend bool operator!=(const Address lhs, const Address rhs);

};

const Address operator"" \_address(const char\* str, size\_t size);

#endif

**Address.cpp**

#include "Address.h"

Address::Address(const std::string &city, const std::string &street, const unsigned int &houseNum, const unsigned int &apartmentNum){

CityName = city;

StreetName = street;

HouseNumber = houseNum;

ApartmentNumber = apartmentNum;

}

Address::Address(const std::string &str){

int position = 0;

while(str[position] != ',' && position < str.length()){

CityName += str[position];

position += 1;

}

position += 1;

while(str[position] != ',' && position < str.length()){

StreetName += str[position];

position += 1;

}

position += 1;

std::string homeStr;

while(str[position] != ',' && position < str.length()){

homeStr += str[position];

position += 1;

}

HouseNumber = std::stoi(homeStr);

position += 1;

std::string apartmentStr;

while(position != str.length()){

apartmentStr += str[position];

position += 1;

}

ApartmentNumber = std::stoi(apartmentStr);

}

std::string Address::IgnoreRegister(const std::string str){

for (int i = 0; i < str.length(); ++i){

tolower(str[i]);

}

return str;

}

void Address::print(){

std::cout << CityName << "," << StreetName << "," << HouseNumber << "," << ApartmentNumber;

}

std::ostream & operator << (std::ostream &out, const Address &address){

out << address.CityName << "," << address.StreetName << "," << address.HouseNumber << "," << address.ApartmentNumber;

return out;

}

void Address::SetAddress(const std::string newCity, const std::string newStreet, const unsigned int newHouseNumber, const unsigned int newApartmentNumber){

CityName = newCity;

StreetName = newStreet;

HouseNumber = newHouseNumber;

ApartmentNumber = newApartmentNumber;

}

bool Address::IsOnThisCity(const std::string City){

return Address::IgnoreRegister(CityName) == Address::IgnoreRegister(City);

}

bool Address::IsOnThisStreet(const std::string City ,const std::string Street){

return (Address::IgnoreRegister(CityName) == Address::IgnoreRegister(City) && Address::IgnoreRegister(StreetName) == Address::IgnoreRegister(Street));

}

bool operator&(const Address lhs, const Address rhs){

return (lhs.CityName == rhs.CityName && lhs.StreetName == rhs.StreetName && abs(lhs.HouseNumber - rhs.HouseNumber) == 1);

}

bool operator<(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber < rhs.ApartmentNumber;

}

else return lhs.HouseNumber < rhs.HouseNumber;

}

else return lhs.CityName < rhs.CityName;

}

else return lhs.CityName < rhs.CityName;

}

bool operator<=(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber <= rhs.ApartmentNumber;

}

else return lhs.HouseNumber <= rhs.HouseNumber;

}

else return lhs.CityName <= rhs.CityName;

}

else return lhs.CityName <= rhs.CityName;

}

bool operator>(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber > rhs.ApartmentNumber;

}

else return lhs.HouseNumber > rhs.HouseNumber;

}

else return lhs.CityName > rhs.CityName;

}

else return lhs.CityName > rhs.CityName;

}

bool operator>=(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber >= rhs.ApartmentNumber;

}

else return lhs.HouseNumber >= rhs.HouseNumber;

}

else return lhs.CityName >= rhs.CityName;

}

else return lhs.CityName >= rhs.CityName;

}

bool operator==(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber == rhs.ApartmentNumber;

}

else return lhs.HouseNumber == rhs.HouseNumber;

}

else return lhs.CityName == rhs.CityName;

}

else return lhs.CityName == rhs.CityName;

}

bool operator!=(const Address lhs, const Address rhs){

if (lhs.CityName == rhs.CityName){

if (lhs.StreetName == rhs.StreetName){

if (lhs.HouseNumber == rhs.HouseNumber){

return lhs.ApartmentNumber != rhs.ApartmentNumber;

}

else return lhs.HouseNumber != rhs.HouseNumber;

}

else return lhs.CityName != rhs.CityName;

}

else return lhs.CityName != rhs.CityName;

}

const Address operator"" \_address(const char\* str, size\_t size){

int position = 0;

std::string city;

while(str[position] != ','){

city += str[position];

position += 1;

}

position += 1;

std::string street;

while(str[position] != ','){

street += str[position];

position += 1;

}

position += 1;

std::string houseStr;

while(str[position] != ','){

houseStr += str[position];

position += 1;

}

int house = std::stoi(houseStr);

position += 1;

std::string apartmentStr;

while(position != size){

apartmentStr += str[position];

position += 1;

}

int apartment = std::stoi(apartmentStr);

Address result(city, street, house, apartment);

return result;

}

**main.cpp**

#include <iostream>

#include "Address.h"

bool CheckFormat(std::string &str){

int position = 0;

while(str[position] != ',' && position < str.length()){

position += 1;

}

if(position == str.length()){

return false;

}

position += 1;

while(str[position] != ',' && position < str.length()){

position += 1;

}

if(position == str.length()){

return false;

}

position += 1;

while(str[position] != ',' && position < str.length()){

if(str[position] < '0' || str[position] > '9'){

return false;

}

position += 1;

}

if(position == str.length()){

return false;

}

position += 1;

while(position != str.length()){

if(str[position] < '0' || str[position] > '9'){

return false;

}

position += 1;

}

return true;

}

int main() {

std::cout << "test literal address:" << "Here's city name,Here's street name,12,34"\_address << "\n";

std::cout << "input address in \"city,street,house,apartment\" format\n";

std::string currentStr;

std::cout << "insert address1:";

std::getline(std::cin, currentStr);

if (!CheckFormat(currentStr)){

std::cout << "invalid format\n";

return 0;

};

Address address1(currentStr);

std::cout << "insert address2:";

std::getline(std::cin, currentStr);

if (!CheckFormat(currentStr)){

std::cout << "invalid format\n";

return 0;

};

Address address2(currentStr);

std::cout << "address1 is ";

address1.print();

std::cout << "\n";

std::cout << "address2 is ";

address2.print();

std::cout << "\n";

std::string locationCity;

std::cout << "insert city name for location of address1 and address2:";

std::getline(std::cin, locationCity);

std::cout << "is address1 located in this city? " << (address1.IsOnThisCity(locationCity) ? "YES" : "NO") << "\n";

std::cout << "is address2 located in this city? " << (address2.IsOnThisCity(locationCity) ? "YES" : "NO") << "\n";

std::string locationStreet;

std::cout << "insert street name for location of address1 and address2:";

std::getline(std::cin, locationStreet);

std::cout << "is address1 located in this street? " << (address1.IsOnThisStreet(locationCity, locationStreet) ? "YES" : "NO") << "\n";

std::cout << "is address2 located in this street? " << (address2.IsOnThisStreet(locationCity, locationStreet) ? "YES" : "NO") << "\n";

std::cout << "\n";

std::cout << "is address1 and address2 are neighbors? " << (address1 & address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 == address2? " << (address1 == address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 != address2? " << (address1 != address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 < address2? " << (address1 < address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 <= address2? " << (address1 <= address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 > address2? " << (address1 > address2 ? "YES" : "NO") << "\n";

std::cout << "is address1 >= address2? " << (address1 >= address2 ? "YES" : "NO") << "\n";

return 0;

}

**Выводы**

Я научился перегружать операторы класса через дружественные функции, реализовывать пользовательские литералы. Также научился перегружать операторы std::ostream для упрощения печати объекта.

**Список литературы**

1. Урок №36. Литералы и магические числа

URL: https://ravesli.com/urok-139-peregruzka-operatora-kruglye-skobki/

(дата обращения: 07.10.2020)

1. Урок №134. Перегрузка операторов через методы класса

URL:https://ravesli.com/urok-134-peregruzka-operatorov-cherez-metody-klassa/

(дата обращения: 07.10.2020)

1. Урок №113. Классы, Объекты и Методы

URL:https://ravesli.com/urok-113-klassy-obekty-i-metody-klassov/

(дата обращения: 11.10.2020)