Московский Авиационный Институт
(Национальный Исследовательский Университет)

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

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
I I I семестр
Задание 1: «Простые классы»

Группа:	М8О-208Б-18, №19
Студент:	Овечкин Виталий Андреевич
Преподаватель:	Журавлёв Андрей Андреевич
Оценка:	
Дата:	28.10.2019

Москва, 2019

1. **Тема**: Простые классы

2. **Цель работы**: Изучение системы сборки на языке C++, изучение систем контроля версии. Изучение основ работы с классами в C++, перегрузка операторов, использование пользовательских литералов.

3. **Задание** (*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.

4. **Адрес репозитория на GitHub** https://github.com/vitalouivi/oop_exercise_02

5. **Код программы на C++**

main.cpp

```
#include "lab2.h"

int main() {
	using namespace lab2;
	std::string city1, street1;

	Address adr1;
	Address adr2;
	Address adr3;
	adr3 = "( Moscow , st1 , 5 , 15 )"_adr;
	std::cout << "Third adress: ";
	adr3.Write(std::cout);
	std::cout << "Enter address 1: ";
	adr1.Read(std::cin);

	std::cout << "First adress: ";
	adr1.Write(std::cout);
	std::cout << "Enter address 2: ";
	adr2.Read(std::cin);

	std::cout << "Second adress: ";
	adr2.Write(std::cout);


	if(adr1 == adr2)
		std::cout << "Address1 is similar to address2" << std::endl;
	else
		std::cout << "Address1 is not similar to address2" << std::endl;
	if(adr1&adr2)
		std::cout << "Address1 is near address2" << std::endl;
	else
		std::cout << "Address1 is not near address2" << std::endl;


	std::cout << "Enter city: ";
	std::cin >> city1;
	std::cout << "Enter street: ";
	std::cin >> street1;
	adr1.Belong(city1, street1, adr1);
	adr2.Belong(city1, street1, adr2);
```

```
}
```

Lab2.h

```
#ifndef D_LAB2_H_
#define D_LAB2_H_ 1

#include <cmath>
#include <cctype>
#include <string>
#include <cstring>
#include <iostream>
#include <sstream>

namespace lab2 {
	class Address
	{
	private:
		std::string city;
		std::string street;
		int house;
		int flat;

	public:
		static std::string ToLower(std::string);

		Address();
		Address(std::string c, std::string s, int h, int f);

		void Read(std::istream& is);
		void Write(std::ostream& os) const;

		std::string City() const;
		std::string Street() const;
		int House() const;
		int Flat() const;

		bool operator==(const Address& adr) const;
		bool operator&(const Address& adr) const;

		void Belong(std::string c, std::string s, const Address& adr) const;

		friend std::istream& operator>> (std::istream& in, Address& adr);
		friend std::ostream& operator<< (std::ostream& out, const Address& adr);
	};
	Address operator""_adr(const char* str, size_t size);
}

#endif
```

Lab2.cpp

```
#include "lab2.h"

namespace lab2 {
	std::string Address::ToLower(std::string s) {
		std::string low = "";

		for (int i = 0; i < (int)s.length(); i++) {
			low += tolower(s[i]);
		}
		return low;
	}

	Address::Address() {
		city = "";
		street = "";
```

```
		house = 0;
		flat = 0;
	}

	Address::Address(std::string c, std::string s, int h, int f) {
		city = c;
		street = s;
		house = h;
		flat = f;
	}

	void Address::Read(std::istream& is) {
		is >> city >> street >> house >> flat;
		city = ToLower(city);
		street = ToLower(street);
	}
	void Address::Write(std::ostream& os) const {
		os << city << " " << street << " " << house << " " << flat << "\n";
	}

	std::string Address::City() const {
		return ToLower(city);
	}
	std::string Address::Street() const {
		return ToLower(street);
	}
	int Address::House() const {
		return house;
	}
	int Address::Flat() const {
		return flat;
	}

	bool Address::operator==(const Address& adr) const {
		if ((this->City() == adr.City()) && (this->Street() == adr.Street()) &&
(this->House() == adr.House()) && (this->Flat() == adr.Flat()))
			return true;
		else
			return false;
	}

	bool Address::operator&(const Address& adr) const {
		if ((this->City() == adr.City()) && (this->Street() == adr.Street()) &&
abs(this->House() - adr.House()) <= 2)
			return true;

		else
			return false;
	}

	void Address::Belong(std::string c, std::string s, const Address& adr) const {
		if (adr.City() == ToLower(c) && adr.Street() == ToLower(s))
			std::cout << "This address belongs" << std::endl;
		else
			std::cout << "This address doesn't belong" << std::endl;
	}

	std::istream& operator>> (std::istream& in, Address& adr) {
		in >> adr.city >> adr.street >> adr.house >> adr.flat;
		return in;
	}
	std::ostream& operator<< (std::ostream& out, const Address& adr) {
		out << adr.city << " " << adr.street << " " << adr.house << " " << adr.flat
<< "\n";
		return out;
```

```
	}

	Address operator""_adr(const char* str, size_t size) {//( city,street,house,flat )
		std::istringstream is(str);
		char tmp;
		std::string c;
		std::string s;
		int h, f;
		is >> tmp >> c >> tmp >> s >> tmp >>h >> tmp >> f ;
		return {c,s,h,f};
	}
}
```

CMakeLists.txt

```
cmake_minimum_required (VERSION 3.5)

project(lab2)

add_executable(oop_exercise_02
                    main.cpp
                lab2.cpp)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra")

set_target_properties(oop_exercise_02 PROPERTIES CXX_STANDART 14
CXX_STANDART_REQUIRED ON)
```

6. Набор testcases

test_01.txt	Ожидаемое действие	Ожидаемый результат
Moscow	По соседству	First adress is near second
Str1	Первый адрес принадлежит улице	First adress belongs
20	Второй адрес принадлежит улице	Second adress belongs
10		
Moscow		
Str1		
20		
15		
Moscow		
Str1		

test_02.txt	Ожидаемое действие	Ожидаемый результат
Msc	Не по соседству	First adress is near second
Str1	Первый адрес не принадлежит улице	First adress doesn’t belong
10	Второй адрес не принадлежит улице	Second adress doesn’t belong
5		
Spb		
St		
5		
15		
spb		
Str		

test_03.txt	Ожидаемое действие	Ожидаемый результат
Msc	Не по соседству	First adress is near second
Str1	Первый адрес не принадлежит улице	First adress doesn’t belong
10	Второй адрес принадлежит улице	Second adress belongs
5		
Spb		
St		
5		
15		
spb		
St		

7. Результаты выполнения тестов

```
MacBook-Air-mac:lab1 mac$ ./lab2
Third adress: Moscow St1 5 15
Enter address 1: Moscow Str1 20 10
First adress: Moscow Str1 20 10
Enter address 2: Moscow str1 20 15
Second adress: Moscow str1 20 15
First adress is near second
Enter city: Moscow
Enter street: str1
First adress belongs
Second adress belongs
MacBook-Air-mac:lab1 mac$ ./lab2
Third adress: Moscow St1 5 15
Enter address 1: Msc str1 10 5
First adress: Msc str1 10 5
Enter address 2: Spb st 5 15
Second adress: Spb st 5 15
First adress isn't near second
Enter city: spb
Enter street: str
First adress doesn't belong
Second adress doesn't belong
MacBook-Air-mac:lab1 mac$ ./lab2
Third adress: Moscow St1 5 15
Enter address 1: Msc str1 10 5
First adress: Msc str1 10 5
Enter address 2: Spb st 5 15
Second adress: Spb st 5 15
First adress isn't near second
Enter city: spb
Enter street: st
First adress doesn't belong
Second adress belongs
```

8. Объяснение результатов работы программы - вывод

В lab2.h были заданы, а в lab2.cpp описаны, методы и свойства этого класса,перегрузки операторов, пользовательский литерал ,применяемые в main.cpp. За основу взята лабораторная работа 1, некоторые ф-ции заменены перегрузкой операторов
Классы, описывают методы и свойства объектов, позволяют работать с этими объектами, не вдаваясь в подробности их реализации, что является примером абстракции данных. Такой подход незаменим при работе в групповых проектах.