**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Савров Никита

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

**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

Для выполнения программы создадим класс Address содержащий поля city,street,house,apartment. Для присвоения значений создадим функцию Set в которую будем передавать нужные значения. Для сравнения двух адресов напишем перегрузку оператора ==, в котором будем сравнивать каждое поле. Так же напишем функции is\_city и is\_street для проверки принадлежности адреса к городу и улице. Для выполнения последнего пункта задания используем перегрузку оператора & для проверки являются ли адреса соседними. В функции main инстанциируем 2 объекта класса Address и будем вызывать к ним нужные функции.

1. Набор testcases

test\_01

1

MoscoW lenina 10 11

2

MOScow leninA 10 11

3

4

Leningrad 1

5

lenina 2

6

0

test\_02

1

MOScow lenina 10 11

2

Leningrad mira 11 12

3

4

Moscow 1

5

lenina 2

6

0

test\_03

1

MosCOw lenina 10 13

2

Moscow lenina 11 14

3

4

Stalingrad 1

5

mira 2

6

0

1. Результаты выполнения тестов.

test\_01

Literal test

Moscow Mira 10 12

Stalingrad Lenina 15 20

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1

MoscoW lenina 10 11

2

MOScow leninA 10 11

3

4

Leningrad 1

5

lenina 2

6

0

Enter Address 1

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

Enter Address 2

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

equal

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter city and number address

this address not locate in Leningrad

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter sreet and number address

this address locate in lenina

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

this addresses are not neighbour

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

test\_02

Literal test

Moscow Mira 10 12

Stalingrad Lenina 15 20

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1

MOScow lenina 10 11

2

Leningrad mira 11 12

3

4

Moscow 1

5

lenina 2

6

0

Enter Address 1

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

Enter Address 2

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

not equal

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter city and number address

this address locate in Moscow

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter sreet and number address

this address locate in lenina

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

this addresses are not neighbour

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

test\_03

Literal test

Moscow Mira 10 12

Stalingrad Lenina 15 20

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1

MosCOw lenina 10 13

2

Moscow lenina 11 14

3

4

Stalingrad 1

5

mira 2

6

0

Enter Address 1

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

Enter Address 2

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

not equal

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter city and number address

this address not locate in Stalingrad

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

enter sreet and number address

this address not locate in mira

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

this addresses are neighbour

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1 - input address 1

2 - input address 2

3 - compare addresses

4 - this address locate in this city

5 - this address locate on this street

6 - are this addresses neighbour

0 - exit

1. Листинг программы

// Лабораторная работа №2 Савров Никита М80-207Б-18

// 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.

#include <iostream>

#include <string>

#include <algorithm>

#include <cmath>

using namespace std;

class Address{

public:

string city;

string street;

unsigned long long int house;

unsigned long long int apartment;

Address(){

city = "";

street = "";

house = 0;

apartment = 0;

}

Address(string str1, string str2, unsigned long long int hous, unsigned long long int home){

city = str1;

street = str2;

house = hous;

apartment = home;

}

void Set(string s1, string s2, unsigned long long int a1, unsigned long long int a2){

city = s1;

street = s2;

house = a1;

apartment = a2;

}

bool is\_city(string town){

transform(town.begin(), town.end(), town.begin(), (int (\*)(int))tolower);

transform(city.begin(), city.end(), city.begin(), (int (\*)(int))tolower);

return city == town;

}

bool is\_street(string stree){

transform(stree.begin(), stree.end(), stree.begin(), (int (\*)(int))tolower);

transform(this->street.begin(), this->street.end(), this->street.begin(), (int (\*)(int))tolower);

return this->street == stree;

}

};

bool operator ==(Address a, Address b){

transform(a.city.begin(), a.city.end(), a.city.begin(), (int (\*)(int))tolower);

transform(b.city.begin(), b.city.end(), b.city.begin(), (int (\*)(int))tolower);

transform(a.street.begin(), a.street.end(), a.street.begin(), (int (\*)(int))tolower);

transform(b.street.begin(), b.street.end(), b.street.begin(), (int (\*)(int))tolower);

return a.city == b.city && a.street == b.street && a.house == b.house && a.apartment == b.apartment;

}

Address operator +(Address a, Address b){

Address result;

result.city = a.city + b.city;

result.street = a.street + b.street;

result.house = a.house + b.house;

result.apartment = a.apartment + b.apartment;

return result;

}

bool operator &(Address a, Address b){

transform(a.city.begin(), a.city.end(), a.city.begin(), (int (\*)(int))tolower);

transform(b.city.begin(), b.city.end(), b.city.begin(), (int (\*)(int))tolower);

transform(a.street.begin(), a.street.end(), a.street.begin(), (int (\*)(int))tolower);

transform(b.street.begin(), b.street.end(), b.street.begin(), (int (\*)(int))tolower);

if (a.city == b.city && a.street == b.street){

if (a.house > b.house)

if (a.house - b.house == 1){

return true;

}

if (a.house < b.house)

if (b.house - a.house == 1){

return true;

}

}

return false;

}

Address operator ""\_cit (const char \*ans, size\_t){

string str(ans);

return Address(str,"",0,0);

}

Address operator ""\_stree (const char \*ans, size\_t){

string str(ans);

return Address("",str,0,0);

}

Address operator ""\_hom (unsigned long long int x){

return Address("","",x,0);

}

Address operator ""\_aps (unsigned long long int y){

return Address("","",0,y);

}

int main(){

Address a("","",0,0);

Address b("","",0,0);

Address c = "Moscow"\_cit + "Mira"\_stree + 10\_hom + 12\_aps;

Address d = "Stalingrad"\_cit + "Lenina"\_stree + 15\_hom + 20\_aps;

cout << "Literal test\n" << c.city << " " << c.street << " " << c.house << " " << c.apartment << "\n";

cout << d.city << " " << d.street << " " << d.house << " " << d.apartment << "\n";

unsigned long long int item = 1;

string name1,name2,town,stree;

unsigned long long int adress;

unsigned long long int name3,name4;

cout << "1 - input address 1\n2 - input address 2\n3 - compare addresses\n4 - this address locate in this city\n5 - this address locate on this street\n6 - are this addresses neighbour\n0 - exit\n";

while (item != 0){

cin >> item;

switch (item) {

case 1:

cout << "Enter Address 1\n";

cin >> name1 >> name2 >> name3 >> name4;

a.Set(name1, name2, name3, name4);

break;

case 2:

cout << "Enter Address 2\n";

cin >> name1 >> name2 >> name3 >> name4;

b.Set(name1, name2, name3, name4);

break;

case 3:

if (a == b)

cout << "equal\n";

else

cout << "not equal\n";

break;

case 4:

cout << "enter city and number address\n";

cin >> town >> adress;

if (adress == 1){

if (a.is\_city(town))

cout << "this address locate in " << town << '\n';

else

cout << "this address not locate in " << town << '\n';

break;

}

if (adress == 2){

if (b.is\_city(town))

cout << "this address locate in " << town << '\n';

else

cout << "this address not locate in " << town << '\n';

break;

}

cout << "this number not exist\n";

break;

case 5:

cout << "enter sreet and number address\n";

cin >> stree >> adress;

if (adress == 1){

if (a.is\_street(stree))

cout << "this address locate in " << stree << '\n';

else

cout << "this address not locate in " << stree << '\n';

break;

}

if (adress == 2){

if (a.is\_street(stree))

cout << "this address locate in " << stree << '\n';

else

cout << "this address not locate in " << stree << '\n';

break;

}

cout << "this number not exist\n";

break;

case 6:

if (a & b){

cout << "this addresses are neighbour\n";

}else

cout << "this addresses are not neighbour\n";

break;

}

cout << "1 - input address 1\n2 - input address 2\n3 - compare addresses\n4 - this address locate in this city\n5 - this address locate on this street\n6 - are this addresses neighbour\n0 - exit\n";

}

}

<https://github.com/trol53/oop_exercise_01/tree/master/oop_exercise_02>

6.Вывод

Данная программа может быть полезна для работы с адресами. Так- же ее можно улучшить добавив дополнительные функции вроде проверки нахождения квартир в одном доме и прочих. Благодаря парадигме ООП мы можем представлять пользовательские типы данных в виде объектов что значительно упрощает программирование. Так-же эта работа показывает практическое применение пользовательских литералов и перегрузки операторов.

Литература

1.Информационный портал хабр[электронный ресурс] URL:<https://habr.com/ru/post/140357/>

(дата обращения 30.09.2019)

2.Информационный портал хабр [электронный ресурс] URL: <https://habr.com/ru/post/132014/>

(дата обращения 30.09.2019)