**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: Савченко Илья Владимирович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

Вариант 19

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

* **address.h**
  + Класс **Address**
    - Приватные члены
      * строчки **city**, **street**, а также числа **house** и **flat**
    - Публичные члены
      * Конструкторы
      * Функция Normalize, которая вызывает функцию MakeName на обе строчки city и street
      * Функция belongsTo, которая проверяет принадлежность к городу и улице
      * Функция AdrComp, которая возвращает 1, 0, -1 в зависимости от сравнения двух адресов
      * Перегрузки операторов сравнения, &, а также вывода
  + функция MakeName, исправляющая регистр входных строк, первая буква заглавная, остальные — строчные
  + Создание пользовательского литерала “”\_adr
* **main.cpp**
  + Программа сравнивает два адреса, “Moskow Pushkina 13 37” и вводимого со стд. входа,  
    после чего проходит демонстрация функций вывода, сравнения и принадлежности.

1. Набор тестов

|  |  |
| --- | --- |
| ==> test\_01.txt <==  MoSkOw PuShKiNa 14 48 | Moskow Pushkina 13 37  Moskow Pushkina 14 48  In Moscow? 1  myHome is in Moscow, Pushkina 1  Adress in Moscow Pushkina? 1  myHome \* friendHome  1 &  1 <  0 > |
| ==> test\_02.txt <==  mosKOW pushKINA 12 25 | Moskow Pushkina 13 37  Moskow Pushkina 12 25  In Moscow? 1  myHome is in Moscow, Pushkina 1  Adress in Moscow Pushkina? 1  myHome \* friendHome  1 &  0 <  1 > |
| ==> test\_03.txt <==  moskow PUSHKINA 10 15 | Moskow Pushkina 13 37  Moskow Pushkina 10 15  In Moscow? 1  myHome is in Moscow, Pushkina 1  Adress in Moscow Pushkina? 1  myHome \* friendHome  0 &  0 <  1 > |
| ==> test\_04.txt <==  moskow sovetskaya 14 41 | Moskow Pushkina 13 37  Moskow Sovetskaya 14 41  In Moscow? 1  myHome is in Moscow, Pushkina 1  Adress in Moscow Pushkina? 0  myHome \* friendHome  0 &  1 <  0 > |
| ==> test\_05.txt <==  Irkutsk Pushkina 12 45 | Moskow Pushkina 13 37  Irkutsk Pushkina 12 45  In Moscow? 0  myHome is in Moscow, Pushkina 1  Adress in Moscow Pushkina? 0  myHome \* friendHome  0 &  0 <  1 > |

1. Результаты выполнения тестов  
   Результаты тестов показывают нам работоспособность всех функций, необходимых для реализации класса **Address**
2. Листинг программы

==> main.cpp <==

/\* Савченко Илья Владимирович

\* М8О-208Б-19

\*

\* 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.

\*/

#include<iostream>

#include<string>

#include"address.h"

using namespace std;

int main(){

Address myHome = "mOskoW puSHkinA 13 37"\_adr;

string newCity,

newStreet;

int newHouse,

newFlat;

cin >> newCity

>> newStreet

>> newHouse

>> newFlat;

Address friendHome(newCity, newStreet, newHouse, newFlat);

cout << '\n';

cout << myHome << '\n'

<< friendHome << '\n';

cout << "In Moscow? "<< friendHome.belongsTo("mOsKow") << '\n';

cout << "myHome is in Moscow, Pushkina " << myHome.belongsTo("MoskOw", "pushkINA") << '\n';

cout << "Adress in Moscow Pushkina? " << friendHome.belongsTo("MOSKOW", "pushkina") << '\n';

cout << "\nmyHome \* friendHome\n"

<< (myHome & friendHome) << " &\n"

<< (myHome < friendHome) << " <\n"

<< (myHome > friendHome) << " >\n";

}

==> address.h <==

#include<string>

#include<iostream>

using namespace std;

void MakeName(string &str){

for(int i=0; str[i]; ++i){

if(str[i] >= 'A' && str[i] <= 'Z')

str[i] += 'a'-'A';

}

str[0] -= 'a'-'A';

}

class Address{

private:

string city;

string street;

int house;

int flat;

public:

Address();

Address(

const string &city,

const string &street,

int house = 1,

int flat = 1

) {

this->city = city;

this->street = street;

this->house = house;

this->flat = flat;

this->Normalize();

}

void Normalize(){

MakeName(city);

MakeName(street);

}

friend ostream& operator<< (ostream &out, const Address &adr){

out << adr.city << " "

<< adr.street << " "

<< adr.house << " "

<< adr.flat;

return out;

}

bool belongsTo(string city, string street = "") const{

MakeName(city);

if(this->city != city)

return false;

if(street.size() == 0)

return true;

MakeName(street);

return this->street == street;

}

// A = B = 0

// A < B = -

// A > B = +

friend int AdrComp(const Address &A, const Address &B){

int temp;

temp = A.city.compare(B.city);

if(temp != 0)

return temp;

temp = A.street.compare(B.street);

if(temp != 0)

return temp;

if(A.house != B.house)

return A.house - B.house;

return A.flat - B.flat;

}

friend bool operator==(const Address &A, const Address &B)

{return AdrComp(A, B) == 0;}

friend bool operator!=(const Address &A, const Address &B)

{return AdrComp(A, B) != 0;}

friend bool operator< (const Address &A, const Address &B)

{return AdrComp(A, B) < 0;}

friend bool operator<=(const Address &A, const Address &B)

{return AdrComp(A, B) <= 0;}

friend bool operator> (const Address &A, const Address &B)

{return AdrComp(A, B) > 0;}

friend bool operator>=(const Address &A, const Address &B)

{return AdrComp(A, B) >= 0;}

friend bool operator& (const Address &A, const Address &B){

if(!A.belongsTo(B.city, B.street))

return false;

int temp = A.house - B.house;

if(temp < 0)

temp = -temp;

return temp == 1;

}

};

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

int i = 0;

string city;

for( ; str[i] != ' '; ++i)

city += str[i];

++i;

string street;

for( ; str[i] != ' '; ++i)

street += str[i];

++i;

int house;

for(house = 0; str[i] != ' '; ++i){

house \*= 10;

house += str[i] - '0';

}

++i;

int flat;

for(flat=0; i<n; ++i){

flat \*= 10;

flat += str[i] - '0';

}

return Address(city, street, house, flat);

}

Литература

1. Уроки программирования на C++ | Ravesli [Электронный ресурс]  
   URL: <https://ravesli.com/uroki-cpp/>  
   (дата обращения: 23.11.2020)