Московский авиационный институт

(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 1

Тема: Простые классы на языке C++

Студент: Короткевич Леонид Витальевич

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

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).

2. Описание программы

main.cpp	
std::string to_lower(const std::string &str)	для сравнения строк без учета регистра
Address()	конструктор по умолчанию
Address(const std::string &new_town, const std::string &new_street, const int &new_apt, const int &new_house)	конструктор с 4 параметрами
bool operator==(Address const &rhs) const	оператор сравнения экземпляров класса
bool inTown(std::string const &aTown) const	константный метод: принадлежность городу
bool inStreet(std::string const &aStreet) const	константный метод: принадлежность улице
bool isNeighs(Address const &rhs) const	константный метод: проверка, является ли соседом
void display() const	константный метод: вывод всей информации адресе

3. Набор тестов

test_01.txt	test_02.txt	test_03.txt
1 msk fest 4 3 1 spb fest 4 3 1 msK fesT 4 3 3 1 2 3 1 3 7 2 2 7 8	1 ekb lenina 123 1 1 EKB LENINa 123 1 1 ekb lenina 123 1 7 6 1 3 6 1 2 1 EKB LENINA 124 0 7 6 1 4 3 1 4 8	1 ufa zelenka 2 3 1 ufa ZELENKA 4 5 1 istra lenina 2 1 4 1 zelenKA 4 1 zelenka 4 2 ZELENKa 5 1 UFA 5 3 IsTrA 6 1 2 8

4. Результаты выполнения тестов

test_01.txt	test_02.txt	test_03.txt
1 - добавить адрес 2 - удалить адрес 3 - сравнить два адреса 4 - проверить принадлежность улице 5 - проверить принадлежность городу 6 - являются ли соседями 7 - вывести список адресов 8 - выйти Адрес msk fest 4 3 успешно добавлен в список Адрес spb fest 4 3 успешно добавлен в список Адрес msK fesT 4 3 успешно добавлен в список Адреса НЕ равны Адреса равны № Город Улица Дом Кватира 1 msk fest 3 4 2 spb fest 3 4 3 msK fesT 3 4 Адрес spb fest 3 4 успешно	1 - добавить адрес 2 - удалить адрес 3 - сравнить два адреса 4 - проверить принадлежность улице 5 - проверить принадлежность городу 6 - являются ли соседями 7 - вывести список адресов 8 - выйти Адрес ekb lenina 123 1 успешно добавлен в список Адрес EKB LENINa 123 1 успешно добавлен в список Адрес ekb lenina 123 1 успешно добавлен в список № Город Улица Дом Кватира 1 ekb lenina 1 123 2 EKB LENINa 1 123 3 ekb lenina 1 123	1 - добавить адрес 2 - удалить адрес 3 - сравнить два адреса 4 - проверить принадлежность улице 5 - проверить принадлежность городу 6 - являются ли соседями 7 - вывести список адресов 8 - выйти Адрес ufa zelenka 2 3 успешно добавлен в список Адрес ufa ZELENKA 4 5 успешно добавлен в список Адрес istra lenina 2 1 успешно добавлен в список Адрес под номером 1 действительно принадлежит улице zelenKA Адрес под номером 1 действительно принадлежит улице zelenka Адрес под номером 2 действительно принадлежит улице ZELENKa Адрес под номером 1 действительно принадлежит городу UFA Адрес под номером 3 действительно принадлежит городу IsTrA

удален из списка № Город Улица Дом Кватира 1 msk fest 3 4 2 msK fesT 3 4	Адреса совпадают Адреса совпадают Адрес EKB LENINA 124 0 успешно добавлен в список № Город Улица Дом Кватира 1 ekb lenina 1 123 2 EKB LENINa 1 123 3 ekb lenina 1 123 4 EKB LENINA 0 124 Адреса под номерами 1 и 4 являются соседними Адреса НЕ равны	Адреса под номерами 1 и 2 НЕ являются соседними

5. Листинг программы

```
/*
Короткевич Л. В.
github.com/anxieuse
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).
*/

#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
#include <stdio.h>

// чтобы сравнивать строки без учета регистра
std::string to_lower(const std::string &str)
{
  std::string copy = str;
  for (auto &ch : copy)
     ch = tolower(ch);
  return copy;
}

class Address
{
public:
  Address() : apt(0), house(0) {}
  Address(const std::string &new_town, const std::string &new_street,
```

```
        const int &new_apt, const int &new_house)
  {
     town = new_town;
     street = new_street;
     apt = new_apt;
     house = new_house;
  }
  bool operator==(Address const &rhs) const
  {
     return to_lower(rhs.town) == to_lower(town) &&
          to_lower(rhs.street) == to_lower(street) &&
          rhs.apt == apt &&
          rhs.house == house;
  }
  // принадлежность городу
  bool inTown(std::string const &aTown) const
  {
     return std::equal(town.begin(), town.end(), aTown.begin(),
                  [](char a, char b) {
                     return tolower(a) == tolower(b);
                  });
  }
  // принадлежность улице
  bool inStreet(std::string const &aStreet) const
  {
     return std::equal(street.begin(), street.end(), aStreet.begin(),
                  [](char a, char b) {
                     return tolower(a) == tolower(b);
                  });
  }
  // является ли соседом
  bool isNeighs(Address const &rhs) const
  {
     if (!(to_lower(rhs.town) == to_lower(town) &&
          to_lower(rhs.street) == to_lower(street)))
        return false;
     return abs(rhs.apt - apt) <= 1;
  }
  // вывести всю информацию об одном адресе
  void display() const
  {
     std::cout << town << "\t" << street << "\t" << house << "\t" << apt;
  }

private:
  std::string town, street; // название города, улицы
  int apt, house;          // номер дома, квартиры
};

int main()
{
  std::string town, street;
  int apt, house, idToDel, id, id1, id2;
```

```
    std::vector<Address> ads;

    int command_idx;
    bool input = true;
    std::cout << '\n'
              << " 1  - добавить адрес" << '\n'
              << " 2  - удалить адрес" << '\n'
              << " 3  - сравнить два адреса" << '\n'
              << " 4  - проверить принадлежность улице" << '\n'
              << " 5  - проверить принадлежность городу" << '\n'
              << " 6  - являются ли соседями" << '\n'
              << " 7  - вывести список адресов" << '\n'
              << " 8  - выйти\n"
              << '\n';
    while (input)
    {
        std::cin >> command_idx;
        switch (command_idx)
        {
            case 1:
            {
                std::cin >> town >> street >> apt >> house;
                    printf("Адрес %s %s %d %d успешно добавлен в список\n", town.c_str(), street.c_str(),
apt, house);
                ads.push_back({town, street, apt, house});
                break;
            }
            case 2:
            {
                std::cin >> idToDel;
                --idToDel;
                if (idToDel < ads.size() && idToDel >= 0)
                {
                    std::cout << "Адрес ";
                    ads[idToDel].display();
                    std::cout << " успешно удален из списка\n";
                    ads.erase(ads.begin() + idToDel);
                }
                else
                {
                    std::cout << "Введен некорректный индекс\n";
                }
                break;
            }
            case 3:
            {
                std::cin >> id1 >> id2;
                if (ads[id1 - 1] == ads[id2 - 1])
                {
                    std::cout << "Адреса равны\n";
                }
                else
                {
```

```
                std::cout << "Адреса НЕ равны\n";
            }
            break;
        }
        case 4:
        {
            std::cin >> id >> street;
            if (ads[id - 1].inStreet(street))
            {
                    printf("Адрес под номером %d действительно принадлежит улице %s\n", id,
street.c_str());
            }
            else
            {
                printf("Адрес под номером %d НЕ принадлежит улице %s\n", id, street.c_str());
            }
            break;
        }
        case 5:
        {
            std::cin >> id >> town;
            if (ads[id - 1].inTown(town))
            {
                    printf("Адрес под номером %d действительно принадлежит городу %s\n", id,
town.c_str());
            }
            else
            {
                printf("Адрес под номером %d НЕ принадлежит городу %s\n", id, town.c_str());
            }
            break;
        }
        case 6:
        {
            std::cin >> id1 >> id2;
            if (ads[id1 - 1].Address::isNeighs(ads[id2 - 1]))
            {
                if (ads[id1 - 1] == ads[id2 - 1])
                    printf("Адреса совпадают\n");
                else
                    printf("Адреса под номерами %d и %d являются соседними\n", id1, id2);
            }
            else
            {
                printf("Адреса под номерами %d и %d НЕ являются соседними\n", id1, id2);
            }
            break;
        }
        case 7:
        {
            if (ads.size() == 0)
            {
                std::cout << "Список адресов пуст\n";
```

```
            }
            else
            {
                printf("\n№\tГород\tУлица\tДом\tКватира\n");
                for (auto &x : ads)
                {
                    std::cout << &x - &ads[0] + 1 << '\t';
                    x.Address::display();
                    std::cout << '\n';
                }
                std::cout << '\n';
            }
            break;
        }
        case 8:
        {
            input = false;
            break;
        }
    }
  }
}
```