Министр науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 3

Анализ данных.

Выполнил студент группы № M3106
Шеин Максим Андреевич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург

2020

Текст задания

Определить:
1. Маршрут с наибольшим количеством остановок по отдельными видам транспорта
2. Наиболее длинный маршрут (основывая на координатах) по отдельным видам транспорта
3. Улицу с наибольшим числом остановок

Для извлечения данных из xml-файла воспользоваться библиотекой pugixml, либо любым аналогом.
Для хранения и подсчета статистики, спроектировать и реализовать структуры данных, обеспечивающие оптимальную алгоритмическую сложность расчетов и не избыточность по памяти.

Решение

```
#include "pugixml.hpp"
#include <iostream>
#include <vector>
#include <algorithm>
#include <sstream>
#include <map>
#include <set>
#include <cmath>
#include<cstring>

using std::vector;
using std::cout;
using std::map;
using std::set;
using std::string;
using std::stringstream;

#define ELEMM Elements(std::stoi(iter.child_value("number")), routes,
iter.child_value("type_of_vehicle"), iter.child_value("object_type"),
iter.child_value("name_stopping"), iter.child_value("the_official_name"), locations,
cords)

//Класс координат
class Point
{

private:
    float x, y;
public:

    Point(const float x_, const float y_)
    {

        this->x = x_;
        this->y = y_;

    }

    float getX() const
    {

        return this->x;

    }

    float getY() const
    {
```

```
        return this->y;

    }

};

//Класс Элементов(Номер маршрута, локация и т.д.)
class Elements
{

private:

    int num;

    string type_of_vehicle, object_type, name_stopping, the_official_name;

    vector<string> locations;

    vector<string> routes;

    Point cords;

public:

    Elements(int number, vector<string> routes, string type_of_vehicle, string
object_type, string the_official_name, string name_stopping, vector<string> location,
Point coord) : type_of_vehicle(move(type_of_vehicle)), num(number), routes(move(routes)),
object_type(move(object_type)), the_official_name(move(the_official_name)),
name_stopping(move(name_stopping)), locations(move(location)), cords(coord){}

    string getTypeOfVehicle() const
    {

        return this->type_of_vehicle;

    }

    float getCoordX() const
    {

        return this->cords.getX();

    }

    float getCoordY() const
    {

        return this->cords.getY();

    }

    int vecSize() const
    {

        return this->routes.size();

    }

    string vecIndex(int index) const
    {

        return this->routes[index];

    }
};
```

```
//Класс маршрутов
class Routes
{

public:
    string route;

    vector<Elements> Tram;

    vector<Elements> Bus;

    vector<Elements> Trolleybus;

};

//Вычисление дистанции
double distCoord(const Elements &a, const Elements &b)
{

    double lat1 = a.getCoordX() * 3.1415926 / 180;
    double long1 = a.getCoordY() * 3.1415926 / 180;
    double lat2 = b.getCoordX() * 3.1415926 / 180;
    double long2 = b.getCoordY() * 3.1415926 / 180;

    double deltaLong = long2 - long1;
    double deltaLat = lat2 - lat1;

    double result = pow(sin(deltaLat / 2), 2) + cos(lat1) * cos(lat2) * pow(sin(deltaLong
/ 2), 2);

    result = 2 * asin(sqrt(result)) * 6371;

    return result;
}

//Парсинг пар, например, для координат
void parsePair(const string &a1, string &a2, string &a3)
{
    string distributor = ",";

    auto start = 0;
    auto end = a1.find(distributor);

    while (end != -1)
    {

        a2 = a1.substr(start, end - start);

        start = end + distributor.length();

        end = a1.find(distributor, start);

    }

    a3 = a1.substr(start, end);
}

//Корректность локации
string correctLocation(string &s)
{
    vector<string> errors{"ул.", " ул.", " УЛ.", " ш.", " Ш.", " шоссе", " ШОССЕ", "
пер.", " ПЕР.", " переулок", " ПЕРЕУЛОК", " улица", "улица", " УЛИЦА", " бул", " БУЛ", "
бульвар", " БУЛЬВАР", " пр", " ПР", " проспект", " ПРОСПЕКТ"};
```

```
    for (auto const& elem : errors)
    {

        if (s.find(elem) != -1)
        {

            unsigned long first = s.find(elem);

            unsigned long second = first + elem.size();

            s.erase(first, second);

        }

    }

    if (s[s.size() - 1] == ' ')
    {

        s.erase(s.end() - 1);

    }

    return s;
}

//Парсинг
void parsingFile(vector<Elements> &Stops, map<string, Routes> &mapRoutes, set<string>
&RoutesNum, map<string, int> &Locations)

{

    pugi::xml_document doc;
    doc.load_file("data.xml");
    pugi::xml_node data = doc.child("dataset");

    for (pugi::xml_node iter = data.child("transport_station"); iter; iter =
iter.next_sibling("transport_station"))
    {

        //Координаты
        string string1, string2;

        parsePair(iter.child_value("coordinates"), string1, string2);

        float FirstCords = stof(string1);
        float SecondCords = stof(string2);

        Point cords = Point(FirstCords, SecondCords);

        //Маршруты
        string1 = string2 = "";

        string RoutesString = iter.child_value("routes"), segment;

        vector<string> routes;

        stringstream tempRoutesString(RoutesString);

        if (count(RoutesString.begin(), RoutesString.end(), ','))
        {

            while (getline(tempRoutesString, segment, ','))
            {
```

```
                routes.push_back(segment);

            }
        }
        else
        {

            while (getline(tempRoutesString, segment, '.'))
            {

                routes.push_back(segment);

            }

        }

        //Локации
        string1 = string2 = "";
        string LocationString = iter.child_value("location");
        vector<string> locations;

        stringstream tempLocationString(LocationString);
        if (count(LocationString.begin(), LocationString.end(), ',') &&
!LocationString.empty())
        {

            while (getline(tempLocationString, segment, ','))
            {

                if (segment[0] == ' ')
                {

                    segment.erase(segment.begin());

                }

                locations.push_back(correctLocation(segment));
                Locations[correctLocation(segment)] += 1;
            }

        }
        else if (!LocationString.empty())
        {

            locations.push_back(correctLocation(LocationString));
            Locations[correctLocation(LocationString)] += 1;

        }

        Stops.emplace_back(ELEMM);

        if (!strcmp(iter.child_value("type_of_vehicle"), "Трамвай"))
        {

            for (auto const &element : routes)
            {

                mapRoutes[element].Tram.emplace_back(ELEMM);
                mapRoutes[element].route = element;
                RoutesNum.insert(element);

            }
        }
        else if (!strcmp(iter.child_value("type_of_vehicle"), "Автобус"))
        {
```

```
            for (auto const &element : routes)
            {

                mapRoutes[element].Bus.emplace_back(ELEMM);
                mapRoutes[element].route = element;
                RoutesNum.insert(element);

            }

        }
        else if (!strcmp(iter.child_value("type_of_vehicle"), "Троллейбус"))
        {

            for (auto const &element : routes)
            {

                mapRoutes[element].Trolleybus.emplace_back(ELEMM);
                mapRoutes[element].route = element;
                RoutesNum.insert(element);

            }

        }

    }

}
//1)массив со всеми станциями и со всей информацией о этой станции
//2)пара номера маршрута и массив траспорта с таким маршрутом
//3)номера всех маршрутов
//4)пара названия локации и частота её появления

int main()
{

    setlocale(LC_ALL, "Russian");

    vector<Elements> Stops; // массив со всеми станциями и со всей информацией об этой
станции
    map<string, Routes> Routes; //пара номера маршрута и массив траспорта с таким
маршрутом
    set<string> NameOfRoutes; //номера всех маршрутов
    map<string, int> Locations; //пара названия локации и частота её появления

    parsingFile(Stops, Routes, NameOfRoutes, Locations);









    ////////////////////////////////////////////////////////////НАИБОЛЬШЕЕ КОЛЛИЧЕСТВО
ОСТАНОВОК МАРШРУТА И НОМЕР ЭТОГО
МАРШРУТА////////////////////////////////////////////////////////////////////////////
    map<string, int> TramRoutes, BusRoutes, TrolleybusRoutes; //пара номера маршрута и
кол-ва его появлений на остановках
```

```
    for (auto const &element: Stops)
    {
        //Трамвай                                (кол-во остановок у номера маршрута)
        if (element.getTypeOfVehicle() == "Трамвай")
        {

            for (int j = 0; j < element.vecSize(); ++j)
            {

                TramRoutes[element.vecIndex(j)] += 1;

            }

        }
        //Автобус                                (кол-во остановок у номера маршрута)
        else if (element.getTypeOfVehicle() == "Автобус")
        {

            for (int j = 0; j < element.vecSize(); ++j)
            {

                BusRoutes[element.vecIndex(j)] += 1;

            }

        }
        //Троллейбус                             (кол-во остановок у номера маршрута)
        else if (element.getTypeOfVehicle() == "Троллейбус")
        {

            for (int j = 0; j < element.vecSize(); ++j)
            {

                TrolleybusRoutes[element.vecIndex(j)] += 1;

            }

        }

    }

    string TramMaxRoutes, BusMaxRoutes, TrolleybusMaxRoutes; // номера маршрутов с
наибольшим количеством остановок

    int TramCountStops = 0, BusCountStops = 0, TrolleybusCountStops = 0; // счётчики
максимальных значений

    //Трамвай                                    (сравниваем кол-во остановок у
маршрутов)
    for (auto const& element : TramRoutes)
    {

        if (element.second > TramCountStops)
        {

            TramCountStops = element.second;
            TramMaxRoutes = element.first;

        }

    }
    //Автобус                                    (сравниваем кол-во остановок у
маршрутов)
    for (auto const& element : BusRoutes)
    {
```

```
        if (element.second > BusCountStops)
        {

            BusCountStops = element.second;
            BusMaxRoutes = element.first;

        }

    }
    //Троллейбус                                    (сравниваем кол-во остановок у
маршрутов)
    for (auto const& element : TrolleybusRoutes)
    {

        if (element.second > TrolleybusCountStops)
        {

            TrolleybusCountStops = element.second;
            TrolleybusMaxRoutes = element.first;

        }

    }









    ////////////////////////////////////////////////////////////САМЫЙ ДЛИННЫЙ
МАРШРУТ//////////////////////////////////////////////////////////////////////////////

    map<string, float> TramRoutesSize, BusRoutesSize, TrolleybusRoutesSize; //пара номера
маршрута и его длины

    for (auto const &name : NameOfRoutes)
    {
        //Трамвай                                   (собираем длину маршрута)
        if (Routes[name].Tram.size() > 1)
        {

            for (int k = 0; k < Routes[name].Tram.size() - 1; k++)
            {

                TramRoutesSize[Routes[name].route] += distCoord(Routes[name].Tram[k],
Routes[name].Tram[k + 1]);

            }
        }
        //Автобус                                   (собираем длину маршрута)
        if (Routes[name].Bus.size() > 1)
        {

            for (int k = 0; k < Routes[name].Bus.size() - 1; k++)
            {

                BusRoutesSize[Routes[name].route] += distCoord(Routes[name].Bus[k],
Routes[name].Bus[k + 1]);
```

```
            }

        }
        //Троллейбус                                (собираем длину маршрута)
        if (Routes[name].Trolleybus.size() > 1)
        {

            for (int k = 0; k < Routes[name].Trolleybus.size() - 1; k++)
            {

                TrolleybusRoutesSize[Routes[name].route] +=
distCoord(Routes[name].Trolleybus[k], Routes[name].Trolleybus[k + 1]);

            }

        }

    }

    string TramMaxRoute, BusMaxRoute, TrolleybusMaxRoute; //Номера самых длиных маршрутов

    float TramConstRoute = 0, BusConstRoute = 0, TBusConstRoute = 0; //Счётчики их длин

    //Трамвай                                   (разделяем пару на номер маршрута / его
длина)
    for (auto const& elem : TramRoutesSize)
    {
        if (elem.second > TramConstRoute)
        {

            TramConstRoute = elem.second;
            TramMaxRoute = elem.first;

        }

    }
    //Автобус                                   (разделяем пару на номер маршрута / его
длина)
    for (auto const& elem : BusRoutesSize)
    {

        if (elem.second > BusConstRoute)
        {

            BusConstRoute = elem.second;
            BusMaxRoute = elem.first;

        }

    }
    //Троллейбус                                (разделяем пару на номер маршрута / его
длина)
    for (auto const& elem : TrolleybusRoutesSize)
    {

        if (elem.second > TBusConstRoute)
        {

            TBusConstRoute = elem.second;
            TrolleybusMaxRoute = elem.first;

        }

    }
```

```
    ////////////////////////////////////////////////////////////НАИБОЛЬШЕЕ КОЛЛИЧЕСТВО
ОСТАНОВОК В КАКОЙ-ТО
ЛОКАЦИИ//////////////////////////////////////////////////////////////////////////////

    string Street; //локация с наибрльшим количеством остановок

    int maxStops = 0; //счётчик остановок

    for (auto const& elem : Locations)
    {

        if (elem.second > maxStops)
        {

            maxStops = elem.second;
            Street = elem.first;

        }

    }









    cout << "
BIP BUP BIP\n";
    cout << "
Calculation in progress\n";
    cout << "                                                                  Loading...
[][][][][][][][][][] 100%\n";
    cout << "The largest number of TRAM stops is equal to: " << TramCountStops << ".\n{\n
Route number " << TramMaxRoutes << "\n}\n\n";
    cout << "The largest number of BUS stops is equal to: " << BusCountStops << ".\n{\n
Route number " << BusMaxRoutes << "\n}\n\n";
    cout << "The largest number of TROLLEYBUS stops is equal to: " <<
TrolleybusCountStops << ".\n{\n   Route number " << TrolleybusMaxRoutes << "\n}\n\n";

    cout << "\n";
    cout << "Longest TRAM route: " << TramMaxRoute << "\n\n";
    cout << "Longest BUS route: " << BusMaxRoute << "\n\n";
    cout << "Longest TROLLEYBUS route: " << TrolleybusMaxRoute << "\n\n";


    cout << "Most street stops on " << Street << " street is equal to " << maxStops <<
"\n\n";
```

```
    cout << "
BIP BIP BUP\n";
    return 0;

}
```