ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Інформатики

**Звіт**

з лабораторної роботи № 2

Виконав Перевірила

ст.гр.ІТІНФ-20-1 Кириченко I. Ю.

Самченко С.О.

Харків 2021

ВАРІАНТ 21

Розробити клас, що представляє компанію.

**Компанія** характеризується: назвою, адресою, видом діяльності, списком відділів, роком заснування.

**Адреса** характеризується: назвою міста, вулиці, номером будинку.

**Відділ** характеризується: назвою, спеціалізацією, кількістю працівників, процентом завантаження.

Розробити методи, які дозволяють:

1. дізнатися назву та адресу компанії

2. дізнатися рік заснування компанії

3. дізнатися середнє завантаження компанії

4. добавити новий відділ з перевіркою по назві

5. видалити відділ

6. знайти найбільш завантажений відділ

7. дізнатися середню кількість працівників у відділі

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

using namespace std;

struct Addres {

string City;

string Street;

int HouseNumber = 0;

};

ostream& operator <<(ostream& out, const Addres& adr);

istream& operator >>(istream& in, Addres& adr);

struct Department {

string DName;

string Specialization;

int CountEmp = 0;

int ActivityPr = 0;

};

ostream& operator <<(ostream& out, const Department& dep);

istream& operator >>(istream& in, Department& dep);

class Company {

private:

string CompanyName;

string Activity;

int Year;

Addres Adr;

int NumbDep;

Department\* Deps;

public:

/\*-----------------------------------------------------------------------------------------------------\*/

Company();

Company(const Company& obj);

Company(const string& CN, const string& act, const int& year, const Addres& adr);

~Company();

/\*-----------------------------------------------------------------------------------------------------\*/

const string& getCompanyName()const;

const string& getActivity()const;

int getYear()const;

const Addres& getAdr()const;

int getNumbDep()const;

void outPutDep(ostream& out)const;

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

void setCompanyName(const string& compname);

void setActivity(const string& activity);

void setYear(int year);

void setAdr(const string& city, const string& street, int house);

/\*-----------------------------------------------------------------------------------------------------\*/

int avarageActivity()const;

bool isExistdDep(string depname)const;

bool addDep(string depname, string specialization, int empcount, int actpr);

bool addDep(const Department& dep);

bool deleteDep(string depname);

Department\* findMostActivDep()const;

int averageEmpCount()const;

/\*-----------------------------------------------------------------------------------------------------\*/

Company& operator =(const Company& obj);

Company& operator +=(const Department& actor);

bool operator <(const Company& arg2) const;

operator int()const;

};

ostream& operator <<(ostream& out, const Company& actor);

istream& operator >>(istream& in, Company& actor);

int main() {

setlocale(LC\_ALL, "ru");

Company company;

cin >> company;

cout << "-------------------------------" << endl;

cout << "Add new department: " << endl;

Department newDep;

cin >> newDep;

company += newDep;

cout << "-------------------------------" << endl;

cout << "Information about the company:\n";

cout << company;

cout << "-------------------------------" << endl;

cout << endl << endl;

cout << "Avarage count of workers: " << company.averageEmpCount() << endl;

cout << "Average workload in the company: " << company.avarageActivity() << endl;

Department\* MostActivDep = company.findMostActivDep();

if (MostActivDep) cout << "The most loaded department:\n" << \*MostActivDep << endl;

string DepName;

cout << "-------------------------------" << endl;

cout << "Enter the name of the department you want to delete:\n";

getline(cin, DepName);

bool result = company.deleteDep(DepName);

if (!result) cout << "There is no such department " << endl;

cout << "-------------------------------" << endl;

cout << "Residual information about the company:\n";

cout << company << endl;

system("pause");

return 0;

}

/\*-----------------------------------------------------------------------------------------------------\*/

ostream& operator <<(ostream& out, const Department& dep)

{

out << "\nDepartment:\t" << dep.DName

<< "\nSpecialization:\t" << dep.Specialization

<< "\nCount of workers =\t" << dep.CountEmp

<< "\nWorkload in the company =\t" << dep.ActivityPr;

return out;

}

istream& operator >>(istream& in, Department& dep)

{

cout << " \* Input department name: ";

getline(in, dep.DName);

cout << " \* Activity: ";

getline(in, dep.Specialization);

cout << " \* Count workers: ";

string strTmp;

getline(in, strTmp);

dep.CountEmp = atoi(strTmp.c\_str());

cout << " \* Activity percent: ";

getline(in, strTmp);

dep.ActivityPr = atoi(strTmp.c\_str());

return in;

}

ostream& operator <<(ostream& out, const Addres& adr)

{

out << " City " << adr.City

<< " Street " << adr.Street

<< " House № " << adr.HouseNumber;

return out;

}

istream& operator >>(istream& in, Addres& adr) {

cout << " \* Input city name: ";

getline(in, adr.City);

cout << " \* input street name: ";

getline(in, adr.Street);

cout << " \* input house: ";

string strTmp;

getline(in, strTmp);

adr.HouseNumber = atoi(strTmp.c\_str());

return in;

}

/\*-----------------------------------------------------------------------------------------------------\*/

Company::Company() :NumbDep(0), Deps(NULL), CompanyName("New"), Activity("Ukrain"), Year(0), Adr() {}

Company::Company(const Company& obj) {

CompanyName = obj.CompanyName;

Activity = obj.Activity;

Year = obj.Year;

Adr = obj.Adr;

NumbDep = obj.NumbDep;

Deps = new Department[NumbDep];

for (int i = 0; i < NumbDep; ++i) {

Deps[i] = obj.Deps[i];

}

}

Company::Company(const string& CN, const string& act, const int& year, const Addres& adr):

NumbDep(0), Deps(NULL), CompanyName(CN), Activity(act), Year(year), Adr(adr)

{}

Company::~Company() {

if (Deps) delete[] Deps;

Deps = NULL;

NumbDep = 0;

}

/\*-----------------------------------------------------------------------------------------------------\*/

const string& Company::getCompanyName() const { return CompanyName; }

const string& Company::getActivity()const { return Activity; };

int Company::getYear() const { return Year; }

const Addres& Company::getAdr() const { return (Adr); }

int Company::getNumbDep() const { return NumbDep; }

void Company::outPutDep(ostream& out) const { for (int i = 0; i < NumbDep; ++i) out << Deps[i] << endl;}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

void Company::setCompanyName(const string& compname) { CompanyName = compname; }

void Company::setActivity(const string& activity) { Activity = activity; }

void Company::setYear(int year) { Year = year; }

void Company::setAdr(const string& city, const string& street, int house) { Adr.City = city; Adr.Street = street; Adr.HouseNumber = house; }

/\*-----------------------------------------------------------------------------------------------------\*/

int Company::avarageActivity() const {

int result = 0;

for (int i = 0; i < NumbDep; ++i)

result += (Deps[i].ActivityPr);

result /= NumbDep;

return result;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

bool Company::isExistdDep(string depname)const {

for (int i = 0; i < NumbDep; ++i) {

if (Deps[i].DName == depname) {

return true;

}

}

return false;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

bool Company::addDep(string depname, string specialization, int empcount, int actpr) {

Department newDep = { depname, specialization, empcount, actpr };

return addDep(newDep);

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

bool Company::addDep(const Department& dep) {

if (isExistdDep(dep.DName)) return false;

Department\* mas = new Department[NumbDep + 1];

for (int i = 0; i < NumbDep; ++i)

mas[i] = Deps[i];

mas[NumbDep] = dep;

delete[]Deps;

Deps = mas;

++NumbDep;

return true;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

bool Company::deleteDep(string depname) {

if (!isExistdDep(depname)) return false;

Department\* mas = new Department[NumbDep - 1];

int j = 0;

for (int i = 0; i < NumbDep; ++i)

{

if (Deps[i].DName == depname)

{

continue;

}

mas[j++] = Deps[i];

}

delete[]Deps;

Deps = mas;

--NumbDep;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

Department\* Company::findMostActivDep()const {

if (NumbDep <= 0) return NULL;

Department\* MostActivDep = Deps;

for (int i = 1; i < NumbDep; ++i)

{

if (Deps[i].ActivityPr > MostActivDep->ActivityPr) MostActivDep = &Deps[i];

}

return MostActivDep;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

int Company::averageEmpCount()const {

int result = 0;

for (int i = 0; i < NumbDep; ++i)

result += (Deps[i].CountEmp);

result /= NumbDep;

return result;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

Company& Company::operator =(const Company& obj) {

if (this == &obj) return (\*this);

this->~Company();

CompanyName = obj.CompanyName;

Activity = obj.Activity;

Year = obj.Year;

Adr = obj.Adr;

NumbDep = obj.NumbDep;

Deps = new Department[NumbDep];

for (int i = 0; i < NumbDep; ++i)

{

Deps[i] = obj.Deps[i];

}

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

Company& Company::operator +=(const Department& dep) {

addDep(dep);

return(\*this);

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

bool Company::operator < (const Company& com2) const {

return (this->Year < com2.Year);

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

Company::operator int()const {

int result = avarageActivity();

return result;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

ostream& operator <<(ostream& out, const Company& comp) {

out << comp.getCompanyName()

<< "\n - kind of activity: " << comp.getActivity()

<< "\n - year of foundation: " << comp.getYear()

<< "\n - address: " << comp.getAdr() << endl;

comp.outPutDep(out);

return out;

}

/\*|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|\*/

istream& operator >>(istream& in, Company& comp) {

string strTmp, strTmp1, strTmp2;

cout << "Input company name: ";

getline(in, strTmp);

comp.setCompanyName(strTmp);

cout << "enter the type of activity: ";

getline(in, strTmp);

comp.setActivity(strTmp);

cout << "enter the year of foundation of the company: ";

getline(in, strTmp);

comp.setYear(atoi(strTmp.c\_str()));

cout << endl;

cout << "enter address:" << endl;

cout << " \* city ";

getline(in, strTmp);

cout << " \* street ";

getline(in, strTmp1);

cout << " \* house № ";

getline(in, strTmp2);

comp.setAdr(strTmp, strTmp1, atoi(strTmp2.c\_str()));

string ansver;

do {

cout << endl;

cout << "enter data about departments: " << endl;

Department dep;

in >> dep;

comp += dep;

cout << "want to add another department? (y/n):";

getline(in, ansver);

} while (ansver.size() != 0 && ansver[0] == 'y');

return in;

}

