ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ

УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Інформатики

**Звіт**

з лабораторної роботи № 5

Виконав Перевірила

ст.гр.ІТІНФ-20-1 Кириченко I. Ю.

Самченко С.О.

Харків 2021

Створити контейнер об'єктів класів, реалізованих в лабораторній роботі 5 (студентів, гуртожитків, викладачів і т.д.) Контейнер реалізувати як вектор. Провести певні дії з цим контейнером. Не застосовувати цикли, використовувати тільки відповідні стандартні алгоритми. Перетворити класи для вирішення даного завдання.

ВАРІАНТ 21 (1)

Необхідні завдання такі:

1. забезпечити для контейнера можливість зчитування з файлу і запису в файл з використанням ітераторів потоків.

2. забезпечити для контейнера сортування за прізвищем студента (це повинно бути сортування за замовчуванням), а також сортування за його середнім балом за останню сесію.

3. винести в окремий вектор всіх студентів, починаючи зі студента із середнім балом вищим, ніж 4

4. зберегти всі середні бали студентів з вектора з кроку 3 в окремому векторі

5. дізнатися, скільки в векторі з кроку 4 значень середнього бала вище заданого

6. знайти середній бал для всіх студентів з використанням accumulate ()

7. перевірити, чи входить вектор, отриманий на кроці 3, у вихідний вектор як підпослідовність.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <algorithm>

#include<functional>

#include<numeric>

#include <iterator>

#include <ctime>

#include <vector>

#include <fstream>

#include <sstream>

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;

vector<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;

vector<Department>::const\_iterator findDep(string depname)const;

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

bool addDep(const Department& dep);

void setDeps(const vector<Department>& actors\_) { Deps = actors\_; };

const vector<Department>& getDeps() const { return Deps; };

bool deleteDep(string depname);

vector<Department>::const\_iterator 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);

bool cmpByDeps(const Company& a, const Company& b)

{

return a.getNumbDep() < b.getNumbDep();

}

bool greater\_3(const Company& a)

{

return a.getNumbDep() > 3;

}

const vector<Department>& getDeps(const Company& m)

{

return m.getDeps();

}

class ListDeps {

vector<Department>& listDeps;

public:

ListDeps(vector<Department>& x) :listDeps(x) {};

void operator ()(const Company& y) {

copy(y.getDeps().begin(), y.getDeps().end(), back\_inserter(listDeps));

}

};

int getActPr(const Department& a) { return a.ActivityPr; }

int avaragAdd(int res, const Company& m) {

res += m.getYear();

return res;

}

bool cmpCompanyByName(const Company& a, const Company& b)

{

return a.getCompanyName() == b.getCompanyName();

}

struct {

int operator()(int x, const Company& m)

{

return x + m.avarageActivity();

}

} avarageAct;

int main() {

setlocale(LC\_ALL, "");

ifstream fin("companys.txt");

if (!fin)

{

cerr << "File not found";

system("pause");

return 0;

}

//1. обеспечить для контейнера возможность считывания из файла и записи в файл с использованием итераторов потоков.

vector<Company> companys;

copy(istream\_iterator<Company>(fin), istream\_iterator<Company>(), back\_inserter(companys));

cout << "-------------------------------\n" << "Task 1: \n";

copy(companys.begin(), companys.end(), ostream\_iterator<Company>(cout, "\n"));

ofstream fout("result.txt");

copy(companys.begin(), companys.end(), ostream\_iterator<Company>(fout, "\n"));

//2. обеспечить для контейнера сортировки по названию компании , а также сортировка по кол-ву отделов

cout << "-------------------------------\n" << "Task 2: \n";

cout << "Sort by name: \n";

sort(companys.begin(), companys.end());

copy(companys.begin(), companys.end(), ostream\_iterator<Company>(cout, "\n"));

cout << endl;

cout << "Sort by number Departments: \n";

sort(companys.begin(), companys.end(), cmpByDeps);

copy(companys.begin(), companys.end(), ostream\_iterator<Company>(cout, "\n"));

//3. вынести в отдельный вектор все компании, начиная с компании, в которой более 3 отделов

cout << "-------------------------------\n" << "Task 3: \n";

vector<Company>::iterator it;

it = find\_if(companys.begin(), companys.end(), greater\_3);

vector<Company> companys\_greater\_3;

copy(it, companys.end(), back\_inserter(companys\_greater\_3));

copy(companys\_greater\_3.begin(), companys\_greater\_3.end(), ostream\_iterator<Company>(cout, "\n"));

//4. сохранить процент нагрузки всех отделов из вектора шага 3 в новом векторе

cout << "-------------------------------\n" << "Task 4: \n";

vector<Department> deps;

for\_each(companys\_greater\_3.begin(), companys\_greater\_3.end(), ListDeps(deps));

vector<int> depsActivity;

transform(deps.begin(), deps.end(), back\_inserter(depsActivity), getActPr);

copy(depsActivity.begin(), depsActivity.end(),

ostream\_iterator<int>(cout, " "));

cout << endl;

//5. oпределить количество элементов со значениями больше заданного в векторе, полученном на шаге 4

cout << "-------------------------------\n" << "Task 5: \n";

cout << "activity > 300 ";

int numAct = count\_if(depsActivity.begin(), depsActivity.end(), bind2nd(not2(less<int>()), 300));

cout << "count = " << numAct << endl;

//6. найти среднюю загруженность для всех компаний с использованием accumulate()

cout << "-------------------------------\n" << "Task 6: \n";

for\_each(companys.begin(), companys.end(), mem\_fun\_ref(&Company::avarageActivity));

int avaragAge = accumulate(companys.begin(), companys.end(), 0, avarageAct);

avaragAge /= companys.size();

cout << "avaragActivity = " << avaragAge << endl;

//7. проверить, входит ли вектор, полученный на шаге 3, в выходной вектор как подпоследовательность

cout << "-------------------------------\n" << "Task 7: \n";

vector<Company>::iterator itsub;

itsub = search(companys.begin(), companys.end(), companys\_greater\_3.begin(), companys\_greater\_3.end(), cmpCompanyByName);

if (itsub != companys.end()) {

cout << "sub vector found: \n";

copy(itsub, companys.end(), ostream\_iterator<Company>(cout, "\n"));

}

Company company;

cin >> company;

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

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

Department newDep;

cin >> newDep;

company += newDep;

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

cout << company;

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

cout << endl << endl;

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

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

vector<Department>::const\_iterator ActivDep = company.findMostActivDep();

if (company.getNumbDep() > 0) cout << "The most loaded department:\n"

<< \*ActivDep << endl;

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

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

cout << company << endl;

system("pause");

return 0;

}

// конструкторы

Company::Company() : CompanyName(""), Activity(""), Year(0), Adr() {}

Company::Company(const Company& obj) {

CompanyName = obj.CompanyName;

Activity = obj.Activity;

Year = obj.Year;

Adr = obj.Adr;

Deps = obj.Deps;

}

Company::Company(const string& CN, const string& act, const int& year, const Addres& adr) : CompanyName(CN), Activity(act), Year(year), Adr(adr){}

Company::~Company() { Deps.clear(); }

// гетеры сетеры

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 Deps.size(); }

void Company::outPutDep(ostream& out) const { copy(Deps.begin(), Deps.end(), ostream\_iterator<Department>(out, "\n")); }

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; }

/\* методы класса \*/

// узнать среднюю нагрузку на компанию

int calcActivity(int a, const Department& dep) {

return a + dep.ActivityPr;

}

int Company::avarageActivity() const {

int result = accumulate(Deps.begin(), Deps.end(), 0, calcActivity);

result /= Deps.size();

return result;

}

// найти отдел

vector<Department>::const\_iterator Company::findDep(string depname)const

{

Department dep = { depname };

vector<Department>::const\_iterator result = find(Deps.begin(), Deps.end(), dep);

return result;

}

// .............................................................

// доюавить новый отдел

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 (find(Deps.begin(), Deps.end(), dep) != Deps.end()) return false;

Deps.push\_back(dep);

return true;

}

// удалить существующий отдел

bool Company::deleteDep(string depname) {

vector<Department>::const\_iterator it = findDep(depname);

if (it == Deps.end()) return false;

Deps.erase(it);

return true;

}

// найти самый нагруженный отдел

bool MostActivDep(const Department& a, const Department& b) {

return (a.ActivityPr < b.ActivityPr);

}

vector<Department>::const\_iterator Company::findMostActivDep()const {

vector<Department>::const\_iterator it = min\_element(Deps.begin(), Deps.end(), MostActivDep);

return it;

}

// посчитать среднее колычество сотрудников в компании

int calcCountEmp(int a, const Department& dep) {

return a + dep.CountEmp;

}

int Company::averageEmpCount()const {

int result = accumulate(Deps.begin(), Deps.end(), 0, calcCountEmp);

result /= Deps.size();

return result;

}

// перегрузка операторов

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;

Deps = obj.Deps;

}

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;

}

bool operator == (const Department& a, const Department& b) {

return (a.DName == b.DName);

}

// операторы вводв/вывода

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)

{

getline(in, dep.DName, ';');

getline(in, dep.Specialization, ';');

string strTmp;

getline(in, strTmp, ';');

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

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;

}

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;

getline(in, strTmp, '.');

comp.setCompanyName(strTmp);

getline(in, strTmp, '.');

comp.setActivity(strTmp);

getline(in, strTmp, '.');

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

cout << endl;

getline(in, strTmp, '.');

getline(in, strTmp1, '.');

getline(in, strTmp2, '.');

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

getline(in, strTmp, '.');

stringstream streamDeps(strTmp);

vector<Department> deps;

copy(istream\_iterator<Department>(streamDeps), istream\_iterator<Department>(), back\_inserter(deps));

comp.setDeps(deps);

return in;

}

-------------------------------

Task 1:

Silpo

- kind of activity:

supermarket chain

- year of foundation: 1998

- address: City

energodar Street molodejna House № 49

Department:

D1

Specialization: sorting of products

Count of workers = 100

Workload in the company = 300

Department:

D2

Specialization: sale of products

Count of workers = 200

Workload in the company = 200

Department:

D3

Specialization: acceptance of products

Count of workers = 160

Workload in the company = 140

Department:

D4

Specialization: transportation of products

Count of workers = 180

Workload in the company = 250

-------------------------------

Task 2:

Sort by name:

Silpo

- kind of activity:

supermarket chain

- year of foundation: 1998

- address: City

energodar Street molodejna House № 49

Department:

D1

Specialization: sorting of products

Count of workers = 100

Workload in the company = 300

Department:

D2

Specialization: sale of products

Count of workers = 200

Workload in the company = 200

Department:

D3

Specialization: acceptance of products

Count of workers = 160

Workload in the company = 140

Department:

D4

Specialization: transportation of products

Count of workers = 180

Workload in the company = 250

Sort by number Departments:

Silpo

- kind of activity:

supermarket chain

- year of foundation: 1998

- address: City

energodar Street molodejna House № 49

Department:

D1

Specialization: sorting of products

Count of workers = 100

Workload in the company = 300

Department:

D2

Specialization: sale of products

Count of workers = 200

Workload in the company = 200

Department:

D3

Specialization: acceptance of products

Count of workers = 160

Workload in the company = 140

Department:

D4

Specialization: transportation of products

Count of workers = 180

Workload in the company = 250

-------------------------------

Task 3:

Silpo

- kind of activity:

supermarket chain

- year of foundation: 1998

- address: City

energodar Street molodejna House № 49

Department:

D1

Specialization: sorting of products

Count of workers = 100

Workload in the company = 300

Department:

D2

Specialization: sale of products

Count of workers = 200

Workload in the company = 200

Department:

D3

Specialization: acceptance of products

Count of workers = 160

Workload in the company = 140

Department:

D4

Specialization: transportation of products

Count of workers = 180

Workload in the company = 250

-------------------------------

Task 4:

300 200 140 250

-------------------------------

Task 5:

activity > 300 count = 1

-------------------------------

Task 6:

avaragActivity = 222

-------------------------------

Task 7:

sub vector found:

Silpo

- kind of activity:

supermarket chain

- year of foundation: 1998

- address: City

energodar Street molodejna House № 49

Department:

D1

Specialization: sorting of products

Count of workers = 100

Workload in the company = 300

Department:

D2

Specialization: sale of products

Count of workers = 200

Workload in the company = 200

Department:

D3

Specialization: acceptance of products

Count of workers = 160

Workload in the company = 140

Department:

D4

Specialization: transportation of products

Count of workers = 180

Workload in the company = 250

ATB.

supermarket chain.

1992.

energodar.

ukrainska.

54.

.

-------------------------------

Add new department:

D1;

sorting of products;

120;

300;

Information about the company:

ATB

- kind of activity:

supermarket chain

- year of foundation: 1992

- address: City

energodar Street

ukrainska House № 54

Department:

D1

Specialization:

sorting of products

Count of workers = 120

Workload in the company = 300

-------------------------------

Avarage count of workers: 300

Average workload in the company: 120

The most loaded department:

Department:

D1

Specialization:

sorting of products

Count of workers = 120

Workload in the company = 300

-------------------------------

Residual information about the company:

ATB

- kind of activity:

supermarket chain

- year of foundation: 1992

- address: City

energodar Street

ukrainska House № 54

Department:

D1

Specialization:

sorting of products

Count of workers = 120

Workload in the company = 300



