

 Технически Университет – София

Курсова работа

по дисциплината Програмни езици

Гр. София Изготвил: Константин Иванов

Дата: 21.01.2014 г.

Да се дефинира клас банкова сметка с член - данни: уникален код (буква и 5 цифри);

име на притежателя (указател към символен низ);

динамични масиви от внесени и изтеглени суми.

Конструктор, деструктор, предефиниране на операция << за извеждане на данни.

Данните да се съхраняват във файл.

Съставете програма, която:

1. Създава файл от обекти – банкови сметки.

2. Извежда имената на притежателите на повече от една банкови сметки, подредени по азбучен ред.

3. Извежда разликите от внесени и изтеглени суми.

4. Записва в нов файл информацията за сметките, за които внесените и изтеглени суми са еднакви.

#include <cstdlib>

#include <iostream>

#include <list>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

class bank\_account {

private:

 char\* code\_;

 char\* holder\_name\_;

 vector<double> commitments\_;

 vector<double> withdraws\_;

public:

 bank\_account() {

 code\_ = "";

 holder\_name\_ = "";

 }

 bank\_account(char\* code, char\* holder\_name) {

 code\_ = code;

 holder\_name\_ = holder\_name;

 }

 ~bank\_account() {

 commitments\_.clear();

 withdraws\_.clear();

 }

 void setCode(char\* code) {

 code\_ = code;

 }

 void setHolderName(char\* holder) {

 holder\_name\_ = holder;

 }

 char\* getCode() {

 return code\_;

 }

 char\* getHolderName() {

 return holder\_name\_;

 }

 void submit(double money) {

 if(money > 0) {

 commitments\_.push\_back(money);

 }

 }

 void withdraw(double money) {

 if(money > 0) {

 withdraws\_.push\_back(money);

 }

 }

 vector<double> getCommitments() {

 return commitments\_;

 }

 vector<double> getWithdraws() {

 return withdraws\_;

 }

 friend ostream& operator<<(ostream &out, const bank\_account &b) // its friendly so that it can access the private variables

 {

 out << "(";

 out << b.code\_;

 out << ")(";

 out << b.holder\_name\_;

 out << ")";

 out << "(";

 for (vector<double>::const\_iterator i = b.commitments\_.begin(); i != b.commitments\_.end(); ++i) {

 out << \*i << " ";

 }

 out << ")";

 out << "(";

 for (vector<double>::const\_iterator i = b.withdraws\_.begin(); i != b.withdraws\_.end(); ++i) {

 out << \*i << " ";

 }

 out << ");";

 out << endl;

 return out;

 }

};

int main(int argc, char \*argv[])

{

 vector<bank\_account> accounts;

 bank\_account a("L23456", "Petko Petkov");

 a.submit(20);

 a.submit(20);

 a.submit(5);

 a.withdraw(40);

 a.submit(50);

 a.withdraw(55);

 bank\_account b("L12345", "Ivan Ivanov");

 b.submit(10);

 b.submit(20);

 b.submit(50);

 b.withdraw(5);

 bank\_account c("L03456", "Pesho Petkov");

 c.submit(10);

 c.submit(20);

 c.submit(500);

 c.withdraw(100);

 bank\_account d("L00456", "Petko Petkov");

 d.submit(20);

 d.submit(20);

 d.submit(5);

 d.withdraw(10);

 d.submit(50);

 bank\_account e("L00456", "Ivan Ivanov");

 e.submit(20);

 e.submit(20);

 e.submit(5);

 e.withdraw(10);

 e.submit(50);

 accounts.push\_back(a);

 accounts.push\_back(b);

 accounts.push\_back(c);

 accounts.push\_back(d);

 accounts.push\_back(e);

 vector<bank\_account> v = accounts;

 // \*\*\*\*\*\*\*\*\*\*\*\*\* създава файл с банкови сметки \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 ofstream out("bank accounts.txt");

 for (vector<bank\_account>::const\_iterator ci = accounts.begin(); ci != accounts.end(); ++ci) {

 bank\_account b = \*ci;

 out << b;

 }

 out.close();

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 // \*\*\*\*\* Извежда имената на картодържателите на повече от една сметка \*\*\*\*\*\*\*\*\*\*\*\*\*\*

 vector<char\*> h;

 for (vector<bank\_account>::const\_iterator ci = accounts.begin(); ci != accounts.end(); ++ci) {

 bank\_account b = \*ci;

 h.push\_back(b.getHolderName());

 }

 for(unsigned i = 0; i < h.size()-1; i++) {

 for(unsigned j = i+1; j < h.size(); j++) {

 if(strcmp(h.at(i), h.at(j))>0) {

 char\* tmp = h.at(i);

 h.at(i) = h.at(j);

 h.at(j) = tmp;

 }

 }

 }

 for(unsigned i = 0; i < h.size()-1; i++) {

 for(unsigned j = i+1; j < h.size(); j++) {

 if(strcmp(h.at(i), h.at(j))==0) {

 cout << h.at(i) << endl;

 }

 }

 }

 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 // \*\*\*\*\* Извежда разликата от внесени и изтеглени суми по всяка сметка\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 for(unsigned j = 0; j < v.size(); j++) {

 bank\_account b = v.at(j);

 double sum = 0;

 for(unsigned i = 0; i < b.getCommitments().size(); i++) {

 sum += b.getCommitments().at(i);

 }

 for(unsigned i = 0; i < b.getWithdraws().size(); i++) {

 sum -= b.getWithdraws().at(i);

 }

 cout << b.getCode() << " " << b.getHolderName() << " has " << sum << endl;

 }

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 // \*\*\*\* Записва в нов файл информацията за сметките който имат еднакви внесени \*\*\*\*\*\*

 // \*\*\*\* и изтеглени суми \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 ofstream out1("null.txt");

 for(unsigned j = 0; j < v.size(); j++) {

 bank\_account b = v.at(j);

 double sum = 0;

 for(unsigned i = 0; i < b.getCommitments().size(); i++) {

 sum += b.getCommitments().at(i);

 }

 for(unsigned i = 0; i < b.getWithdraws().size(); i++) {

 sum -= b.getWithdraws().at(i);

 }

 if(sum == 0)

 out1 << b;

 }

 out1.close();

 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 system("PAUSE");

 return EXIT\_SUCCESS;

}