Курсова Работа

Изготвил: Юлиан Матеев



ФИТ, ИСИИ, Група 20б

Факултетен №361223011

проверил:

гл. ас. д-р инж. Елена Калчева-Йовкова

Проектирайте и реализирайте приложение ,което служи като система за управление на банкови сметки – дава възможност за прилагане на различни банкови операции и отпечатва резултат след въвеждане на информацията.

1.Дефинирайте клас Account(Сметка) с полета за: номер на сметка; име на банка; тип на сметка(Разплащателна/Спестовна), декларирани с private достъп.

a)Добавете конструктори, деструктор, методи за достъп, методи за отпечатване на информация.

2.Дефинирайте клас PersonalAccount(Лична сметка) като наследник на класа Account с име на притежателя на сметка и баланс(вложени пари) с private достъп.

а) Добавете конструктори, деструктор, методи за достъп;

б)Предефинирайте метод, който отпечатва пълната информация за сметка.

3.Декларирайте функция main() ,в която се обработва информация за сметки:

а)Въведете данни за N сметки (N <=50 се въвежда от клавиатурата), като въвеждате данните за всяка сметка от клавиатурата и ги въвеждате в масив.

б)Отпечатайте на екрана информацията за всички сметки.

в)Отпечатайте на екрана информация за сметките на дадено лице, чието име се въвежда от клавиатурата, или съобщение, че лицето няма открити сметки.

г)Изчислете и отпечатайте на екрана броя на разплащателните и броя на спестовните сметки и общата сума в тях

#include <iostream>

#include <string>

using namespace std;

class Account { //Начало на дефиниция на клас Account

private:

string banknumber;

string bankname;

string vidsmetka;

public:

Account() { // default конструктор

banknumber = "";

bankname = "";

vidsmetka = "";

}

Account(string, string, string); //конструктор с 3 аргумента

void setBanknumber(string bn) { banknumber = bn; }; //методи за достъп

void setBankname(string b) { bankname = b; };

void setVidsmetka(string vs) { vidsmetka = vs; };

string getBanknumber() { return banknumber; };

string getBankname() { return bankname; };

string getVidsmetka() { return vidsmetka; };

void display(); //член-функция display()

~Account() { // деструктор

cout << "Destroying object!!!\n";

}

}; //край на декларация на клас Account

Account::Account(string bn, string b, string vs) { //конструктор с 3 аргумента

banknumber = bn; bankname = b; vidsmetka = vs;

};

void Account::display() //дефиниция на член-функция display()

{

cout << " ,banknumber:" << banknumber << " ,bankname:" << bankname << " ,account type:" << vidsmetka << "\n";

}

class PersonalAccount :public Account { //Начало на дефиниция на клас PersonalAccount наследник на клас Account

private:

string name;

double balance;

public:

PersonalAccount() { // default конструктор

name = "";

balance = 0.0;

}

PersonalAccount(string bn, string b, string vs, string n, double ba); //конструктор с 5 аргумента

void setname(string n) { name = n; }; //методи за достъп

string getname() { return name; };

void setbalance(double ba) { balance = ba; };

double getbalance() { return balance; };

void display(); //предефинирана функция display()

};//край на дефиниция на клас PersonalAccount

PersonalAccount::PersonalAccount(string bn, string b, string vs, string n, double ba) : Account(bn, b, vs) { //конструктор с 5 аргумента

name = n; balance = ba;

}

void PersonalAccount::display() { //дефиниция на предефинираната функция display()

cout << "Account holder:" << name << " "; Account::display(); cout << "balance:" << balance << endl;

}

int main() //главна функция

{

PersonalAccount\* pArr = new PersonalAccount[50]; //динамичен масив

string bn, b, vs, n; double ba, ob1 = 0.0, ob2 = 0.0, ob3; int i, counter1 = 0, counter2 = 0;

cout << "Enter data about accounts\n";

for (int i = 0; i < 50; i++) //цикъл чрез който се въвежда информация в масива

{

cout << "Enter banknumber: "; getline(cin, bn);

pArr[i].setBanknumber(bn);

cout << "Enter bankname: "; getline(cin, b);

pArr[i].setBankname(b);

do { //цикъл който не позволява да се въвеждат различни видове сметки от разплащателна и спестовна

cout << "Enter account type: ";

getline(cin, vs);

if (vs.compare("current") != 0 && vs.compare("savings") != 0) {

cout << "Enter a valid account type!\n";

}

} while (vs != "current" && vs != "savings");

pArr[i].setVidsmetka(vs);

cout << "Enter account holder name: "; getline(cin, n);

pArr[i].setname(n);

do //цикъл който не позволява да се въвеждат отрицателни стойности за баланс

{

cout << "Enter balance: ";

cin >> ba;

if (ba < 0) {

cout << "Enter a valid ammount!\n";

}

} while (ba < 0);

pArr[i].setbalance(ba);

cin.ignore();

}

cout << "Display data about the accounts:\n";

for (i = 0; i <50; i++) //цикъл който отпечатва информацията въведена в масива

{

pArr[i].display();

}

int wth = 0;

cout << "Search by account holder`s name:\n"; getline(cin,n);

for (i = 0; i <50; i++) //цикъл който отпечатва информация за сметките на потърсено лице

{

if (pArr[i].getname().compare(n) == 0) {

pArr[i].display();

wth++;

if (pArr[i].getname().compare(n) != 0)

break;

}

else

continue;

}

if (!wth) //съобщение ако даденото лице няма открита сметка

cout << n << " has not opened an account!\n";

for (i = 0; i <50; i++)//цикъл който намира броя на разплащателните и спестовните сметки и общата сума в тях

{

if (pArr[i].getVidsmetka().compare("current") == 0) {

counter1++;

ob1 += pArr[i].getbalance();

}

else

{

counter2++;

ob2 += pArr[i].getbalance();

}

}

//отпечатване на броя на разплащателните и спестовните сметки и общата сума в тях:

cout << "Number of current accounts is:" << counter1 << " Sum of the balance inside every current account:" << ob1 << endl;

cout << "Number of savings accounts is:" << counter2 << " Sum of the balance inside every savings account:" << ob2 << endl;

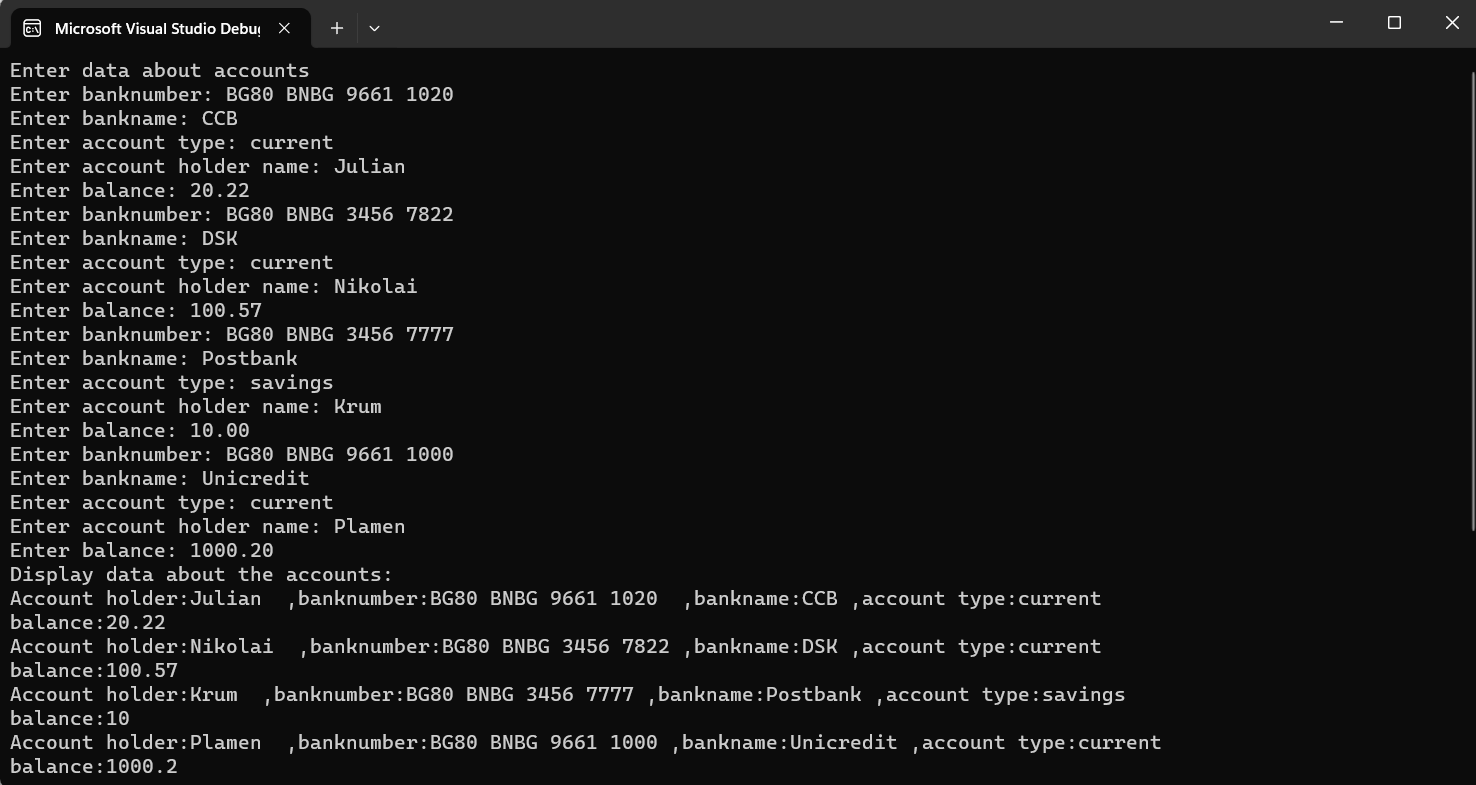
ob3 = ob1 + ob2;

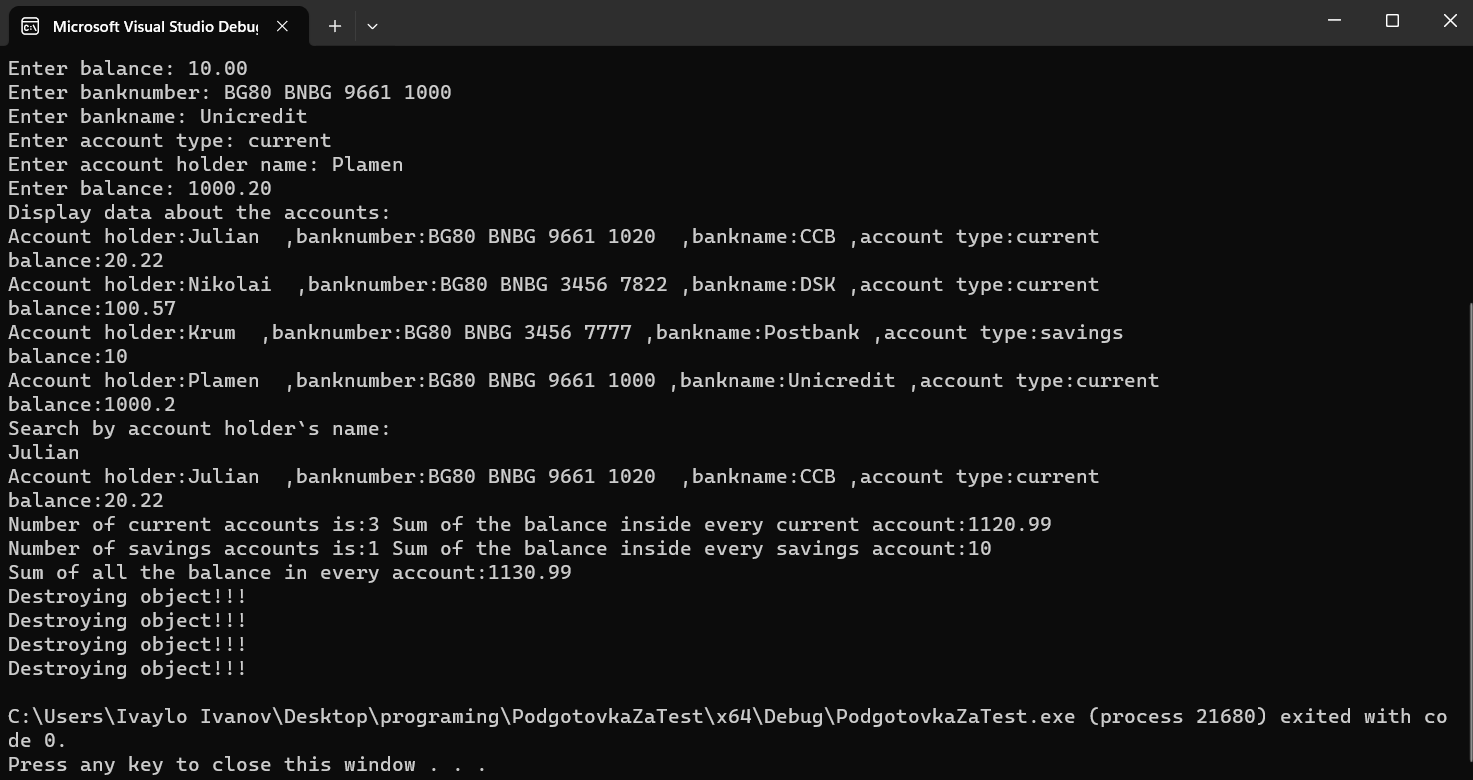
cout << "Sum of all the balance in every account:" << ob3 << endl;

delete[]pArr;

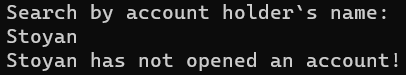
return 0;

}

Тестване на програмата

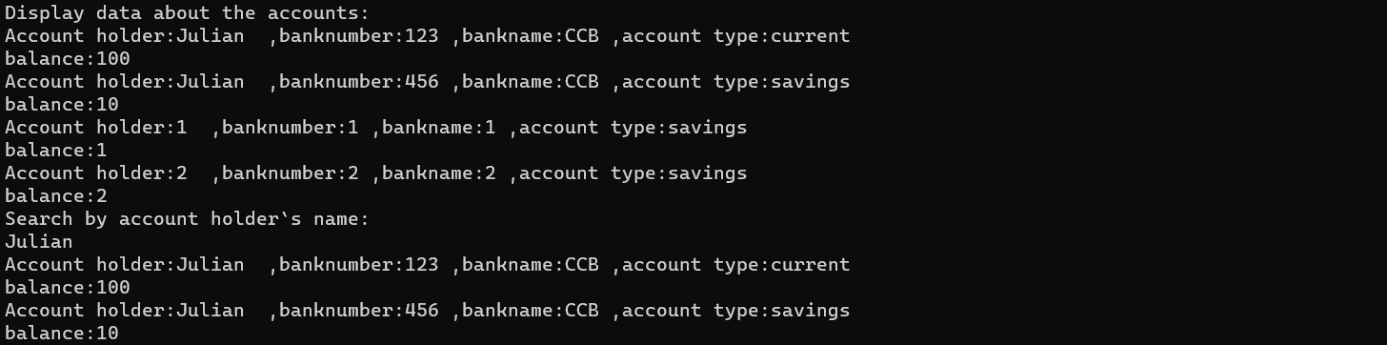


Въвеждане на име на лице което няма открита сметка



Въвеждане на сметка различна от разплащателна и спестовна и отрицателна стойност за баланс



Отпечатване на сметките на лице с повече от една сметка