**Príklad 4.2**

*Zadanie (na samostatnú prácu)*

Rozšírte zdrojový kód riešiaci zadanie príkladu 4.1 tak, aby program vytvorený z tohto rozšíreného zdrojového kódu dokázal taktiež načítať dáta jedného alebo viacerých zamestnancov, tzn. ich meno, priezvisko, rodné číslo, osobné číslo, pracovné zaradenie, názov prevádzky, číslo prevádzky a mzdu. Aby toto Váš rozšírený program dokázal, musíte vytvoriť nové členské funkcie v triede *Zamestnanec*, ktoré budú vedieť zmeniť hodnoty chránených inštančných premenných *osobne\_cislo, prac\_zaradenie* a *nazov\_prevadzky* objektu tejto triedy.

Pre každého z načítaných zamestnancov si program vytvorí zvláštny **objekt** v **dynamickom poli objektov**, čím bude narábať so svojim pamäťovým priestorom veľmi efektívne.

Program bude schopný zobraziť inštančné dáta každého z vytvorených objektov triedy *Zamestnanec* na konzolu.

Program nebude vytvárať a inicializovať objekt triedy *Zamestnanec* pomocou parametrického konštruktora tejto triedy.

Triedy.h

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* definicie tried programu \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class RodneCislo

{

private:

char rodcis[12];

int pohlavie; // 0 - zena, 1 - muz

public:

RodneCislo() {}

//explicitny parametricky konstruktor

RodneCislo(char \*r);

char \*VratRC();

int VratDen();

int VratMesiac();

int VratRok();

int VratPohlavie();

void ZmenRC(char \*r);

//Operatorova funkcia prepisaneho operatora bitoveho posuvu pre triedu 'RodneCislo' musi byt

//deklarovana so specifikatorom pristupu 'friend', aby mohol prepisany operator pristupovat

//k sukromnym datovym clenom triedy 'RodneCislo'.

//Tato operatorova funkcia musi vratit referenciu na prud, ktory dostava ako vstupny parameter,

//pretoze inak by nefungovalo "retazenie" prepisanych operatorov '<<'

friend ostream& operator<<(ostream &vyst\_prud, RodneCislo r);

};

class Osoba

{

private:

char meno[20], priezvisko[20];

public:

//Pretoze vo funkcii 'main' vytvarame dynamicke pole objektov typu 'Osoba',

//a k jeho inicializacii sa default-ovo pouzije bezparametricky konstruktor triedy 'Osoba',

//ktory musi existovat, tak ho tu musime vytvorit

//a navazne nan aj bezparametricky konstruktor triedy 'RodneCislo', pretoze jej objekt je

//zakomponovany do triedy 'Osoba'

Osoba() {}

~Osoba() {} //bezparametricky destruktor

//vytvorenie objektu 'rc' triedy 'RodneCislo' v triede 'Osoba', cim sa stane objekt triedy 'Osoba'

//zlozenym objektom, skladajucim sa aj z objektu 'rc' (kompozicia objektov)

RodneCislo rc;

//explicitny parametricky konstruktor

Osoba(char \*, char \*, char \*);

char \*VratMeno();

char \*VratPriezvisko();

void ZmenMeno(char \*);

void ZmenPriezvisko(char \*);

//Operatorova funkcia prepisaneho operatora bitoveho posuvu pre triedu 'Osoba' musi byt

//deklarovana so specifikatorom pristupu 'friend', aby mohol prepisany operator pristupovat

//k sukromnym datovym clenom triedy 'Osoba'.

//Tato operatorova funkcia musi vratit referenciu na prud, ktory dostava ako vstupny parameter,

//pretoze inak by nefungovalo "retazenie" prepisanych operatorov '<<'

friend ostream& operator<<(ostream &vyst\_prud, Osoba o);

};

class Zamestnanec :public Osoba

{

private:

long int mzda, osobne\_cislo; //doplnena nova clenska premenna

char prac\_zaradenie[30], nazov\_prevadzky[30]; //doplnene nove clenske premenne

long int cis\_prevadzky;

public:

//explicitny parametricky konstruktor

Zamestnanec() {}

~Zamestnanec() {}

Zamestnanec(char \*, char \*, char \*, long int, char \*, char \*, int, long int);

char \*VratPracZaradenie(); //doplnena nova clenska funkcia

char \*VratNazovPrevadzky(); //doplnena nova clenska funkcia

long int VratOsobneCislo(); //doplnena nova clenska funkcia

int VratCis\_prevadzky();

long int VratMzdu();

void ZmenCis\_prevadzky( long int);

void ZmenMzdu(long int);

void ZmenPracZaradenie(char \*);

void ZmenNazovPrevadzky(char \*);

void ZmenOsobnecislo(long int);

//Operatorova funkcia prepisaneho operatora bitoveho posuvu pre triedu 'Zamestnanec' musi byt

//deklarovana so specifikatorom pristupu 'friend', aby mohol prepisany operator pristupovat

//k sukromnym datovym clenom triedy 'Zamestnanec'.

//Tato operatorova funkcia musi vratit referenciu na prud, ktory dostava ako vstupny parameter,

//pretoze inak by nefungovalo "retazenie" prepisanych operatorov '<<'

friend ostream& operator<<(ostream &vyst\_prud, Zamestnanec z);

};

Triedy.cpp

#include "triedy.h"

/\*\*\*\*\*\*\*\*\*\*\* definicie clenskych funkcii triedy 'RodneCislo' \*\*\*\*\*\*\*\*\*\*\*\*/

//definicia parametrickeho konstruktora triedy 'RodneCislo'

RodneCislo::RodneCislo(char \*r)

{

strcpy(rodcis, r);

pohlavie = VratPohlavie();

}

char \*RodneCislo::VratRC()

{

char \*pom;

pom = new char[12];

strcpy(pom, rodcis);

return pom;

}

int RodneCislo::VratDen()

{

char pom[3];

pom[0] = rodcis[4];

pom[1] = rodcis[5];

pom[2] = '\0';

return (atoi(pom));

}

int RodneCislo::VratMesiac()

{

char pom[3];

pom[0] = rodcis[2];

pom[1] = rodcis[3];

pom[2] = '\0';

return (VratPohlavie() == 0) ? (atoi(pom) - 50) : atoi(pom);

}

int RodneCislo::VratRok()

{

char pom[3];

int int\_pom;

pom[0] = rodcis[0];

pom[1] = rodcis[1];

pom[2] = '\0';

int\_pom = atoi(pom);

if (int\_pom < 15)

return int\_pom + 2000; //2000 - 2014

else

return int\_pom + 1900; //1915 - 1999

}

int RodneCislo::VratPohlavie()

{

char pom[3];

pom[0] = rodcis[2];

pom[1] = rodcis[3];

pom[2] = '\0';

return (atoi(pom) > 50) ? 0 : 1;

}

void RodneCislo::ZmenRC(char \*r)

{

strcpy(rodcis, r);

pohlavie = VratPohlavie();

}

/\*\*\*\*\*\*\*\*\*\*\* definicie clenskych funkcii triedy 'Osoba' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//definicia parametrickeho konstruktora triedy 'Osoba'

Osoba::Osoba(char \*m, char \*p, char \*r) :rc(r)

{

strcpy(meno, m);

strcpy(priezvisko, p);

}

char \*Osoba::VratMeno()

{

char \*pom;

pom = new char[20];

strcpy(pom, meno);

return pom;

}

char \*Osoba::VratPriezvisko()

{

char \*pom;

pom = new char[20];

strcpy(pom, priezvisko);

return pom;

}

void Osoba::ZmenMeno(char \*m)

{

strcpy(meno, m);

}

void Osoba::ZmenPriezvisko(char \*p)

{

strcpy(priezvisko, p);

}

/\*\*\*\*\*\*\*\*\*\*\* definicie clenskych funkcii triedy 'Zamestnanec' \*\*\*\*\*\*\*\*\*\*\*\*/

//definicia parametrickeho konstruktora triedy 'Zamestnanec'

Zamestnanec::Zamestnanec(char \*m, char \*p, char \*r, long int osob\_cis, char \*prac\_zarad,

char \*nazov\_prevadz, int cis\_pr, long int mz) : Osoba(m, p, r)

{

osobne\_cislo = osob\_cis;

strcpy(prac\_zaradenie, prac\_zarad);

strcpy(nazov\_prevadzky, nazov\_prevadz);

cis\_prevadzky = cis\_pr;

mzda = mz;

}

long int Zamestnanec::VratOsobneCislo()

{

return osobne\_cislo;

}

char \*Zamestnanec::VratPracZaradenie()

{

char \*pom;

pom = new char[30];

strcpy(pom, prac\_zaradenie);

return pom;

}

char \*Zamestnanec::VratNazovPrevadzky()

{

char \*pom;

pom = new char[30];

strcpy(pom, nazov\_prevadzky);

return pom;

}

int Zamestnanec::VratCis\_prevadzky()

{

return cis\_prevadzky;

}

long int Zamestnanec::VratMzdu()

{

return mzda;

}

void Zamestnanec::ZmenCis\_prevadzky( long int cis\_p)

{

cis\_prevadzky = cis\_p;

}

void Zamestnanec::ZmenMzdu(long int m)

{

mzda = m;

}

void Zamestnanec::ZmenPracZaradenie(char \*m)

{

strcpy(prac\_zaradenie, m);

}

void Zamestnanec::ZmenNazovPrevadzky(char \*m)

{

strcpy(nazov\_prevadzky, m);

}

void Zamestnanec::ZmenOsobnecislo(long int m)

{

osobne\_cislo = m;

}

//defin. neclensk. spriatelenej operatorovej funkcie prepisaneho operatora << pre obj. tr. 'RodneCislo'

ostream& operator<<(ostream &vyst\_prud, RodneCislo r)

{

vyst\_prud << "datum narodenia: " << r.VratDen() << '.' << r.VratMesiac() << '.'

<< r.VratRok();

return vyst\_prud;

}

//definicia neclensk. spriatelenej operatorovej funkcie prepisaneho operatora << pre objekt tr. 'Osoba'

ostream& operator<<(ostream &vyst\_prud, Osoba o)

{

vyst\_prud << "meno : " << o.VratMeno() << endl

<< "priezvisko : " << o.VratPriezvisko() << endl

<< "rodne cislo : " << o.rc.VratRC() << endl;

//pre vypisanie objektu 'rc' do vystupneho prudu sa zavola operatorova funkcia prepisaneho

//operatora << pre objekt triedy 'RodneCislo'

vyst\_prud << o.rc << endl

<< "pohlavie : ";

char pom[5];

(o.rc.VratPohlavie() == 0) ? strcpy(pom, "zena") : strcpy(pom, "muz");

vyst\_prud << pom << endl;

return vyst\_prud;

}

//defin. neclensk. spriatelenej operatorovej funkcie prepisaneho operat. << pre obj. tr. 'Zamestnanec'

ostream& operator<<(ostream &vyst\_prud, Zamestnanec z)

{

vyst\_prud << "meno : " << z.VratMeno() << endl

<< "priezvisko : " << z.VratPriezvisko() << endl

<< "rodne cislo : " << z.rc.VratRC() << endl;

//pre vypisanie objektu 'rc' do vystupneho prudu sa zavola operatorova funkcia prepisaneho operatora //<< pre objekt triedy 'RodneCislo'

vyst\_prud << z.rc << endl

<< "pohlavie : ";

char pom[5];

(z.rc.VratPohlavie() == 0) ? strcpy(pom, "zena") : strcpy(pom, "muz");

vyst\_prud << pom << endl

<< "osobne cislo : " << z.VratOsobneCislo() << endl //pridane

<< "prac. zaradenie : " << z.VratPracZaradenie() << endl //pridane

<< "nazov prevadzky: " << z.VratNazovPrevadzky() << endl //pridane

<< "cislo prevadzky : " << z.VratCis\_prevadzky() << endl

<< "mzda [Eur] : " << z.VratMzdu() << endl;

return vyst\_prud;

}

Main.cpp

#include "triedy.h"

int main()

{

char m[100], priezv[100], rc[100], nazov\_prevadzky[100], pracovne\_zaradenie[100];

int pocet, i;

int long osobne\_cislo, mzda ,cislo\_prevadzky;

cout << "kolko rodnych cisiel chcete skumat? ";

cin >> pocet;

i = 0;

while (i < pocet)

{

cout << endl;

if (pocet == 1)

{

cout << "vlozte rodne cislo: ";

i++;

}

else

cout << "vlozte " << ++i << ". rodne cislo: ";

cin >> rc;

RodneCislo r(rc); //vytvorenie obj. 'r' tr. 'RodneCislo' pomocou volania parametrickeho konstruktora

cout << endl;

cout << r;

cout << endl << "pohlavie : ";

char pom[5];

(r.VratPohlavie() == 0) ? strcpy(pom, "zena") : strcpy(pom, "muz");

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

}

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

cout << endl << "kolko osob chcete vkladat? ";

cin >> pocet;

//vytvorenie dynamickeho pola s poctom 'pocet' objektov typu 'Osoba',

//k jeho inicializacii sa pouzije bezparametricky konstruktor triedy 'Osoba', ktory musi existovat

Osoba \*p\_osoba = new Osoba[pocet];

i = 0;

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

while (i < pocet)

{

int local\_i = i;

if (pocet == 1)

{

cout << "vlozte meno osoby : ";

cin >> m;

cout << "vlozte priezvisko osoby : ";

cin >> priezv;

cout << "vlozte rodne cislo osoby: ";

cin >> rc;

}

else

{

local\_i++;

cout << endl << "vlozte meno " << local\_i << ". osoby : ";

cin >> m;

cout << "vlozte priezvisko " << local\_i << ". osoby : ";

cin >> priezv;

cout << "vlozte rodne cislo " << local\_i << ". osoby: ";

cin >> rc;

}

//pomocou adresovej aritmetiky '(p\_osoba+i)' sa dostaneme, napr. k prvku pola 'Osoba[pocet]'

//s indexom 1, ak je i==1, cize k objektu s indexom 1 ulozenemu v tomto poli

(p\_osoba + i)->ZmenMeno(m);

(p\_osoba + i)->ZmenPriezvisko(priezv);

(p\_osoba + i)->rc.ZmenRC(rc);

cout << endl << "(data objektu tr. 'Osoba' ulozene v prvku dynamickeho pola 'Osoba[pocet]', ku ktoremu sme pristupili pomocou ukazovatela na tento prvok) " << endl;

//dereferencia ukazovatela '(p\_osoba+i)'

cout << \*(p\_osoba + i) << "---------------------------------------------------------" << endl;

i++;

}

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

cout << endl << "kolko zamestnancov chcete vkladat? ";

cin >> pocet;

Zamestnanec \*p\_zamestnanec = new Zamestnanec[pocet];

//vytvorenie dynamickeho pola s poctom 'pocet' objektov typu 'Osoba',

//k jeho inicializacii sa pouzije bezparametricky konstruktor triedy 'Osoba', ktory musi existovat

i = 0;

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

while (i < pocet)

{

int local\_i = i;

if (pocet == 1)

{

cout << "vlozte meno zamestnanca : ";

cin >> m;

cout << "vlozte priezvisko zamestnanca : ";

cin >> priezv;

cout << "vlozte rodne cislo zamestanca : ";

cin >> rc;

cout << "vlozte Osobne cislo Zamestnanca :";

cin >> osobne\_cislo;

cout << "vlozte pracovne zaradenie Zamestnanca :";

cin >> pracovne\_zaradenie;

cout << "vlozte nazov prevadzky Zamastnanca : ";

cin >> nazov\_prevadzky;

cout << "vlozte cislo prevazky Zamestnanca :";

cin >> cislo\_prevadzky;

cout << "vlozte mzdu Zamestnanca :";

cin >> mzda;

}

else

{

local\_i++;

cout << endl << "vlozte meno " << local\_i << ". Zamestnanca : ";

cin >> m;

cout << "vlozte priezvisko " << local\_i << ". Zamestnanca : ";

cin >> priezv;

cout << "vlozte rodne cislo " << local\_i << ". Zamestnanca : ";

cin >> rc;

cout << "vlozte Osobne cislo " << local\_i << ". Zamestnanca :";

cin >> osobne\_cislo;

cout << "vlozte pracovne zaradenie " << local\_i << ". Zamestnanca :";

cin >> pracovne\_zaradenie;

cout << "vlozte nazov prevadzky " << local\_i << ". Zamastnanca :";

cin >> nazov\_prevadzky;

cout << "vlozte cislo prevazky " << local\_i << ". Zamestnanca :";

cin >> cislo\_prevadzky;

cout << "vlozte mzdu " << local\_i << ". Zamestnanca";

cin >> mzda;

}

//pomocou adresovej aritmetiky '(p\_osoba+i)' sa dostaneme, napr. k prvku pola 'Osoba[pocet]'

//s indexom 1, ak je i==1, cize k objektu s indexom 1 ulozenemu v tomto poli

(p\_zamestnanec + i)->ZmenMeno(m);

(p\_zamestnanec + i)->ZmenPriezvisko(priezv);

(p\_zamestnanec + i)->rc.ZmenRC(rc);

(p\_zamestnanec + i)->ZmenOsobnecislo(osobne\_cislo);

(p\_zamestnanec + i)->ZmenPracZaradenie(pracovne\_zaradenie);

(p\_zamestnanec + i)->ZmenNazovPrevadzky(nazov\_prevadzky);

(p\_zamestnanec + i)->ZmenCis\_prevadzky(cislo\_prevadzky);

(p\_zamestnanec + i)->ZmenMzdu(mzda);

cout << endl << "(data objektu tr. 'Osoba' ulozene v prvku dynamickeho pola 'Osoba[pocet]', ku ktoremu sme pristupili pomocou ukazovatela na tento prvok) " << endl;

//dereferencia ukazovatela '(p\_osoba+i)'

cout << \*(p\_zamestnanec + i) << "---------------------------------------------------------" << endl;

i++;

}

//zmazanie 1-rozmerneho dynamickeho pola, na ktore ukazuje ukazovatel 'p\_osoba', z pamate

delete[] p\_osoba;

char men[] = "Martin"; char p[] = "Mato"; char rcs[] = "840609/4578"; char pozicia[] = "lakyrnik"; char pracov[] = "Lakovna";

cout << endl << "programom vytvoreny a inicializovany objekt pomocou parametrickeho konstruktora triedy 'Zamestnanec': " << endl;

//vytvorenie objektu 'Mato' triedy 'Zamestnanec' pomocou volania parametrickeho konstruktora

Zamestnanec Mato(men, p, rcs, 7000256, pozicia, pracov, 25, 1950);

cout << Mato;

return 0;

}