**Príklad 2.2**

*Zadanie (na samostatnú prácu)*

Rozšírte zdrojový kód riešiaci zadanie príkladu 2.1 o

* verejnú členskú operátorovú funkciu

*Zlomky* ***operator -*** *(Zlomky z)*

, ktorá prepíše operátor - tak, aby sa dalo vykonať odčítanie dvoch zlomkov uložených v objektoch ***z1*** a ***z2*** triedy ***Zlomky*** pomocou jednoduchého výrazu ***z1 - z2***,

* verejnú členskú operátorovú funkciu

*Zlomky Zlomky::****operator /*** *(Zlomky z)*

, ktorá prepíše operátor / tak, aby sa dalo vykonať delenie dvoch zlomkov uložených v objektoch ***z1*** a ***z2*** triedy ***Zlomky*** pomocou jednoduchého výrazu ***z1 / z2***

Príklad vstupu a výstupu takéhoto programu

Program vykona operacie so zlomkami

Vlozte 1. zlomok

Vlozte citatel : 5

Vlozte menovatel: 4

1.zlomok je : 1 1/4

Vlozte 2. zlomok

Vlozte citatel : 6

Vlozte menovatel: 8

2.zlomok je : 3/4

Vysledky:

1 1/4 + 3/4 = 2

1 1/4 - 3/4 = 1/2

1 1/4 \* 3/4 = 15/16

1 1/4 / 3/4 = 1 2/3

Ciselne hodnoty 1. a 2. zlomku:

1 1/4 = 1.25

3/4 = 0.75

//triedy.cpp

#include<iostream>

using namespace std;

//deklaracia neclenskej funkcie

int NSD(int a, int b);

class Zlomky {

private:

int citatel;

int menovatel;

void ZjednodusZlomok(Zlomky &zlomok);

public:

//explicitny parametricky konstruktor

Zlomky(int cit = 0, int men = 1)

{

citatel = cit;

if (menovatel == 0)

menovatel = 1;

else

menovatel = men;

}

//definicie 3 inline clenskych funkcii (nemaju prology)

void Citaj\_zlomok() { cin >> \*this; }

void Vypis\_zlomok() { cout << \*this; }

float Hodnota\_zlomku() { return float(citatel) / float(menovatel); }

void Test\_nuly\_v\_men();

Zlomky operator+(Zlomky z);

Zlomky operator-(Zlomky z);//deklaracia clenskej operatorovej funkcie prepisaneho operatora +

//Operatorove funkcie prepisanych operatorov \* a bitoveho posuvu pre triedu 'Zlomky' musia byt

//deklarovane so specifikatorm pristupu 'friend', aby mohli prepisane operatory pristupovat

//k sukromnym datovym clenom triedy 'Zlomky'.

//V tejto operatorovej funkcii nemusime pouzit parametre s referenciami, pretoze v tejto funkcii

//nemenime instancne data objektov 'zlomok1' a 'zlomok2'

friend Zlomky operator \* (Zlomky zlomok1, Zlomky zlomok2);

friend Zlomky operator / (Zlomky zlomok1, Zlomky zlomok2);

//Tieto operatorove neclenske funkcie musia vratit referenciu na prud, ktory dostavaju ako vstupny

//parameter, pretoze inak by nefungovalo "retazenie" prepisanych operatorov '<<' a '>>'.

friend ostream& operator << (ostream &prud, Zlomky z);

friend istream& operator >> (istream &prud, Zlomky &z);

};

Zlomky.cpp

void Zlomky::Test\_nuly\_v\_men()

{

if (menovatel == 0)

{

cout << "Zlomok obsahuje nulu v menovateli." << endl;

exit(1); //okamzite skoncenie programu

}

}

//Definicia clenskej operatorovej funkcie prepisaneho binarneho operatora + s jednym parametrom 'z'.

//Jej druhym parametrom je ukazovatel 'this' na aktualnu instanciu triedy.

//Objekt s ukazovatelom 'this' posiela spravu sam sebe.

Zlomky Zlomky::operator +(Zlomky z)

{

int cit\_vysledok, men\_vysledok;

men\_vysledok = menovatel \* z.menovatel;

cit\_vysledok = citatel \* z.menovatel + z.citatel \* menovatel;

// volanie parametrickeho konstruktora, ktory vytvori objekt triedy 'Zlomky'

return Zlomky(cit\_vysledok, men\_vysledok);

}

//Definicia operatorovej priatelskej funkcie k triede 'Zlomky', ktora prepisuje operator \*.

//Tato operatorova funkcia nie je clenskou funkciou triedy 'Zlomky', je len priatelskou funkciou tejto //triedy.

Zlomky operator \* (Zlomky z1, Zlomky z2)

{

int cit\_vysledok, men\_vysledok;

cit\_vysledok = z1.citatel \* z2.citatel;

men\_vysledok = z1.menovatel \* z2.menovatel;

// volanie parametrickeho konstruktora, ktory vytvori objekt triedy 'Zlomky'

return Zlomky(cit\_vysledok, men\_vysledok);

}

Zlomky Zlomky::operator -(Zlomky z)

{

int cit\_vysledok, men\_vysledok;

men\_vysledok = menovatel \* z.menovatel;

cit\_vysledok = citatel \* z.menovatel - z.citatel \* menovatel;

return Zlomky(cit\_vysledok, men\_vysledok);

}

Zlomky operator / (Zlomky z1, Zlomky z2)

{

int cit\_vysledok, men\_vysledok;

cit\_vysledok = z1.citatel \* z2.menovatel;

men\_vysledok = z1.menovatel \* z2.citatel;

// volanie prametrickeho konstruktora, ktory vytvori objekt triedy 'Zlomky'

return Zlomky(cit\_vysledok, men\_vysledok);

}

//Definicia neclenskej spriatelenej operatorovej funkcie prepisaneho operatora <<

ostream& operator << (ostream &prud, Zlomky z)

{

int cela\_cast, zvysok;

z.ZjednodusZlomok(z);

z.Test\_nuly\_v\_men();

cela\_cast = z.citatel / z.menovatel;

if (cela\_cast)

{

prud << cela\_cast << " ";

zvysok = z.citatel % z.menovatel;

if (zvysok)

prud << zvysok << "/" << z.menovatel;

}

else

{

prud << z.citatel;

if (z.citatel) // ak je 'z.citatel == 0', napiseme len ju a nie napr. 0/1

prud << "/" << z.menovatel;

}

return prud;

}

//Definicia neclenskej spriatelenej operatorovej funkcie prepisaneho operatora >>.

//Operatorovej funkcii musime odovzdat referenciu na objekt z, pretoze funkcia meni jeho instancne //premenne

istream& operator >> (istream &prud, Zlomky &z)

{

cout << "Vlozte citatel : ";

prud >> z.citatel;

cout << "Vlozte menovatel: ";

prud >> z.menovatel;

return prud;

}

//zdroj.cpp

#include "Zlomky.h"

int main()

{

cout << "Program vykona operacie so zlomkami" << endl << endl;

//pri vytvarani objektov z1, z2 a z3 je volany konstruktor triedy Zlomky bez parametrov, clenom tejto

//triedy je vsak konstruktor Zlomky(int cit = 0, int men = 1) s implicitnymi parametrami a ich

//implicitnymi hodnotami, ktory je v takomto pripade zavolany

Zlomky z1, z2, z3;

cout << "Vlozte 1. zlomok " << endl;

z1.Citaj\_zlomok();

//cin >> z1; //FUNKCNA alternativa

cout << "1.zlomok je : ";

z1.Vypis\_zlomok();

cout << endl << endl;

//cout << "1.zlomok je : " << z1 << endl << endl; //FUNKCNA alternativa

cout << "Vlozte 2. zlomok " << endl;

cin >> z2;

//z2.Citaj\_zlomok(); //FUNKCNA alternativa

cout << "2.zlomok je : ";

z2.Vypis\_zlomok();

cout << endl << endl;

//cout << "2.zlomok je : " << z2 << endl << endl; //FUNKCNA alternativa

cout << "Vysledky: " << endl;

cout << z1 << " + " << z2 << " = " << z1 + z2 << endl;

cout << z1 << " - " << z2 << " = " << z1 - z2 << endl;

cout << z1 << " \*" << z2 << " = " << z1 \* z2 << endl;

cout << z1 << " / " << z2 << " = " << z1 / z2 << endl;

cout << "Ciselne hodnoty 1. a 2. zlomku: " << endl;

cout << z1 << " = " << z1.Hodnota\_zlomku() << endl;

cout << z2 << " = " << z2.Hodnota\_zlomku() << endl;

return 0;

}