**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Технічний коледж**

**НУ «Львівська політехніка»**

**Відділення**

**Інформаційних технологій**

**та комп’ютерної техніки**

**Звіт до:**

Лабораторної роботи № 2

з предмету «НП з ООП»

**Тема:**

«**Перевантаження операторів С++.**»

**Підготував**

Студент групи – 31-ПЗ

Дячок Остап

**Львів 2021**

**Мета роботи:** Ознайомитись з поняттям та призначенням дружніх функцій та конструктора копіювання. Навчитися реалізовувати перевантаження операторів.

**Постановка задачі:**

**Варіант 8**

Описати клас, що реалізовує вказаний нижче тип даних. Клас повинен містити множину конструкторів для створення об'єктів певного типу (конструктор по замочуванню та з параметрами, конструктор копії) та подані у таблиці операції над об’єктами класу (плюс обов’язково операції присвоювання та порівняння) з використанням механізму перевантаження операцій:



**Код :**

[main.cpp](https://www.onlinegdb.com/online_c++_compiler#editor_1) :

#include <iostream>

#include"fraction.h"

using namespace std;

int

main ()

{

fraction a (1, 2);

fraction b (1, 2);

cout << "Fraction a:" << endl;

a.print ();

cout << "Fraction b:" << endl;

b.print ();

cout << "a<=b:";

if (a <= b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a>=b:";

if (a >= b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a<b:";

if (a < b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a>b:";

if (a > b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a==b:";

if (a == b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a!=b:";

if (a != b)

cout << "true" << endl;

else

cout << "false" << endl;

cout << "a\*b:";

(a \* b).print ();

cout << "a-b:";

(a - b).print ();

cout << "a+=b:";

a += b;

a.print ();

cout << "a-=b:";

a -= b;

a.print ();

cout << "a\*=b:";

a \*= b;

a.print ();

cout << "a/=b:";

a /= b;

a.print ();

}

[fraction.cpp](https://www.onlinegdb.com/online_c++_compiler#_editor_8188653561):

#include"fraction.h"

#include <stdexcept>

#include <iostream>

#include <cstdlib>

fraction::fraction ()

{

numer = 1;

denum = 1;

}

fraction::fraction (int n, int d)

{

if (d == 0)

throw std::runtime\_error ("zero division error");

numer = n;

denum = d;

}

fraction fraction::operator\* (fraction & DR2)

{

fraction result ((this->numer \* DR2.numer), (this->denum \* DR2.denum));

return result;

}

fraction fraction::operator- (fraction & DR2)

{

if (this->denum != DR2.denum)

{

fraction

result (((this->numer \* DR2.denum) - (this->denum \* DR2.numer)),

(this->denum \* DR2.denum));

return result;

}

else

{

fraction result ((this->numer - DR2.numer), this->denum);

return result;

}

}

bool fraction::operator> (fraction & DR2)

{

if ((this->numer / this->denum) > (DR2.numer / DR2.denum))

return true;

else

return false;

}

bool fraction::operator>= (fraction & DR2)

{

if ((this->numer / this->denum) >= (DR2.numer / DR2.denum))

return true;

else

return false;

}

bool fraction::operator< (fraction & DR2)

{

if ((this->numer / this->denum) < (DR2.numer / DR2.denum))

return true;

else

return false;

}

bool fraction::operator<= (fraction & DR2)

{

if ((this->numer / this->denum) <= (DR2.numer / DR2.denum))

return true;

else

return false;

}

fraction fraction::operator+= (fraction & DR2)

{

if (this->denum != DR2.denum)

{

this->numer = ((this->numer \* DR2.denum) + (this->denum \* DR2.numer));

this->denum \*= DR2.denum;

return \*this;

}

else

{

this->numer += DR2.numer;

return \*this;

}

}

fraction fraction::operator\*= (fraction & DR2)

{

this->numer \*= DR2.numer, this->denum \*= DR2.denum;

return \*this;

}

fraction fraction::operator/= (fraction & DR2)

{

float

bufer;

bufer = DR2.numer;

DR2.numer = DR2.denum;

DR2.denum = bufer;

this->numer \*= DR2.numer, this->denum \*= DR2.denum;

return \*this;

}

fraction fraction::operator-= (fraction & DR2)

{

if (this->denum != DR2.denum)

{

this->numer = ((this->numer \* DR2.denum) - (this->denum \* DR2.numer));

this->denum \*= DR2.denum;

return \*this;

}

else

{

this->numer -= DR2.numer;

return \*this;

}

}

bool fraction::operator!= (fraction & DR2)

{

if ((this->numer / this->denum) != (DR2.numer / DR2.denum))

return true;

return false;

}

bool fraction::operator== (fraction & DR2)

{

if ((this->numer / this->denum) == (DR2.numer / DR2.denum))

return true;

return false;

}

double

fraction::getnum ()

{

return this->numer;

}

double

fraction::getden ()

{

return this->denum;

}

double

fraction::number ()

{

return numer / (double) denum;

}

void

fraction::print ()

{

std::cout << numer << "/" << denum << std::endl;

}

[fraction.h](https://www.onlinegdb.com/online_c++_compiler#_editor_8192303411):

class fraction

{

private:

float numer, denum;

public:

fraction ();

fraction (int n, int d);

fraction operator\* (fraction & DR2);

fraction operator- (fraction & DR2);

bool operator> (fraction & DR2);

bool operator>= (fraction & DR2);

bool operator< (fraction & DR2);

bool operator<= (fraction & DR2);

bool operator!= (fraction & DR2);

bool operator== (fraction & DR2);

fraction operator+= (fraction & DR2);

fraction operator-= (fraction & DR2);

fraction operator\*= (fraction & DR2);

fraction operator/= (fraction & DR2);

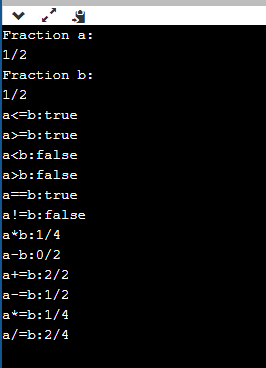
double getnum ();

double getden ();

double number ();

void print ();

};



**Висновок:**

Завдяки виконанню лабораторної роботи 2 я ознайомився з поняттям та призначенням дружніх функцій та конструктора копіювання. Навчився реалізовувати перевантаження операторів.