***ЛАБОРАТОРНА РОБОТА №3***

**Тема:** Перевантаження операцій класу   
**Мета:** Ознайомитись зі способами перевантаження операцій та навчитись використовувати їх при роботі з об’єктами

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.**   
 1. Ознайомитись зі способами перевантаження операцій у С++.   
 2. Проаналізувати приклад програми, яка оголошує клас та перевантажує низку операцій над точками у тривимірному просторі. Визначити результат її роботи.   
 3. В класі Int, який розроблений в завданні №1 лабораторної роботи №1, перевизначте чотири цілочисельні арифметичні операції («+», «-», «\*» , «/») так, щоб їх можна було використовувати для операцій з об'єктами класу Int. Якщо результат будь-якої з операцій виходить за межі типу int (в 32-бітній системі), що може мати значення від 2 147 483 648 до -2 147 483 648, то операція повинна послати повідомлення про помилку і завершити програму. Такі типи даних корисні там, де помилки можуть бути викликані арифметичним переповненням, яке неприпустимо. Напишіть програму для перевірки цього класу. Підказка: для полегшення перевірки переповнення виконуйте обчислення з використанням типу long double.

Код програми:

#include <QCoreApplication>

#include <iostream>

#include <string>

using namespace std;

class Int{

private:

int value;

public:

Int(int defaultVal)

{

value = defaultVal;

}

Int():Int(0){}

void Set(int val) { value = val; }

int Get() { return value; }

void Show() { cout << value << endl; }

void ResetInt() { value = 0; }

void Exception(string str)

{

cout << "Exception: " << str << " in class \"Int\"" << endl;

}

Int operator+(Int n)

{

long double nToCheck = value;

if(value + n.value != nToCheck + n.value)

{

Exception("value overflow");

return 0;

}else return value + n.value;

}

Int operator-(Int n)

{

long double nToCheck = value;

if(value - n.value != nToCheck - n.value)

{

Exception("value overflow");

return 0;

}else return value - n.value;

}

Int operator\*(Int n)

{

long double nToCheck = value;

if(value \* n.value != nToCheck \* n.value)

{

Exception("value overflow");

return 0;

}else return value \* n.value;

}

Int operator/(Int n)

{

long nToCheck = value;

if(value / n.value != nToCheck / n.value)

{

Exception("value overflow");

return 0;

}else return value / n.value;

}

};

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

{

QCoreApplication a(argc, argv);

Int n1(2);

Int n2(10);

(n1+n2).Show();

n1.Set(2147483647);

(n1+n2).Show();

n1.Set(45);

n1 = n1 - n2;

n1.Show();

n2 = n1\*n2;

n2.Show();

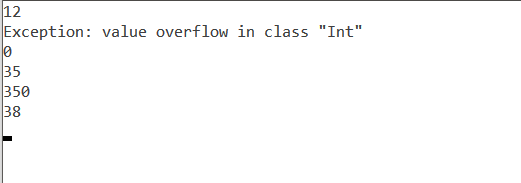
n2 = n2/Int(9);

n2.Show();

return a.exec();

}

Вивід програми:



**Завдання 2.** Для заданого варіанта індивідуального завдання виконати перевантаження операцій для зручності роботи з об’єктами. При необхідності оголосіть певні операторні функції друзями класу.

Код програми:

#include <QCoreApplication>

#include <string.h>

using namespace std;

class Rectangle

{

double x, y;

public:

Rectangle():Rectangle(0,0){}

Rectangle(double \_x, double \_y)

{

SetCornerDigonal(\_x,\_y);

}

~Rectangle(){}

double GetX() {return x;}

double GetY() {return y;}

void SetCornerDigonal(double \_x, double \_y){ x = \_x; y = \_y;}

string GetRectangle(){ return ("Rectangle diagonal is (" + to\_string(x) + "; " + to\_string(y)); }

double GetPerymetr(){ return 2\*x+2\*y; }

Rectangle& operator+(Rectangle& r)

{

Rectangle newR(x+r.x, y+r.y);

return newR;

}

Rectangle& operator-(Rectangle& r)

{

Rectangle newR(x-r.x, y-r.y);

return newR;

}

bool operator>(Rectangle& r)

{

return GetPerymetr() > r.GetPerymetr();

}

bool operator<(Rectangle& r)

{

return GetPerymetr() < r.GetPerymetr();

}

bool operator==(Rectangle& r)

{

return GetPerymetr() == r.GetPerymetr();

}

friend istream& operator<<(istream& stream, Rectangle& r);

friend ostream& operator>>(ostream& stream, Rectangle& r);

};

istream& operator>>(istream& stream, Rectangle& r)

{

double x, y;

stream >> x >> y;

r.SetCornerDigonal(x,y);

return stream;

}

ostream& operator<<(ostream& stream, Rectangle& r)

{

stream << r.GetRectangle();

return stream;

}

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

{

QCoreApplication a(argc, argv);

return a.exec();

}

Висновок: Ознайомився зі способами перевантаження операцій та навчився використовувати їх при роботі з об’єктами.