**ЛАБОРАТОРНА РОБОТА №3**

**Тема:**Перевантаження операцій класу

**Мета:**ознайомитись зі способами перевантаження операцій та навчитись використовувати їх при роботі з об’єктами.

Хід роботи

**Завдання 1**. Ознайомитись зі способами перевантаження операцій у С++.

**Завдання 2**. Проаналізувати приклад програми, яка оголошує клас та перевантажує низку операцій над точками у тривимірному просторі. Визначити результат її роботи.

**Завдання 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.

#include <iostream>

using namespace std;

class Int {

private:

int Inte;

public:

Int(long double a){

if(a>2147483684 || a<-2147483684){

cout<<"error"<<endl;

}

else{

Inte = (int)a;

}

}

void setInt(int \_Int) {

Inte = \_Int;

}

int getInt() {

return Inte;

}

void set0() {

Inte = 0;

}

int sum(int num1, int num2)

{

return num1 + num2;

}

int operator +(Int& a)

{

return Inte+a.Inte;

}

int operator -(Int& a)

{

return Inte-a.Inte;

}

int operator \*(Int& a)

{

return Inte\*a.Inte;

}

double operator /(Int& a)

{

return ((double (Inte))/a.Inte);

}

};

int main()

{

Int a(54), b(2);

Int c = a+b;

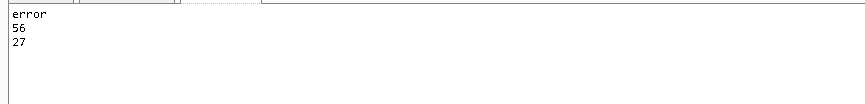
Int d(300000000000);

cout<<c.getInt()<<endl<<a/b<<endl;

return 0;

}

Результат:



**Завдання 4**. Для заданого варіанта індивідуального завдання виконати перевантаження операцій для зручності роботи з об’єктами. При необхідності оголосіть певні операторні функції друзями класу.

#include <iostream>

#include <assert.h>

using namespace std;

class complex

{

double re, im;

public:

complex(double=0,double=0);

~complex();

complex operator +(complex&);

complex operator -(complex&);

complex operator \*(complex&);

complex operator /(complex&);

complex operator ^(unsigned);

friend istream& operator >>(istream&,complex&);

friend ostream& operator <<(ostream&,complex&);

};

complex::complex(double r, double i)

{

re=r; im=i;

}

complex::~complex()

{

}

complex complex::operator +(complex& y)

{

return complex(re+y.re, im+y.im);

}

complex complex::operator -(complex& y)

{

return complex(re-y.re, im-y.im);

}

complex complex::operator \*(complex& y)

{

return complex(re\*y.re-im\*y.im, re\*y.im+im\*y.re);

}

complex complex::operator /(complex& y)

{

double r1=re;

double i1=im;

double r2=y.re;

double i2=y.im;

return complex((r1\*r2-i1\*i2)/(r2\*r2+i2\*i2),(-r1\*i2+i1\*r2) /(r2\*r2+i2\*i2));

{

complex y(1,0);

}

for(unsigned i=1;i<=n;i++)

y=y\*(\*this);

return y;

}

iostream& operator >>(istream& is, complex& x)

{

char c;

is>>x.re;

cin>>c;

assert(c==',');

is>>x.im;

return is;

}

ostream& operator <<(ostream& os, complex& x)

{

os<<x.re<<','<<x.im<<endl;

return os;

}

**Висновок:** ознайомився зі способами перевантаження операцій та навчивсь використовувати їх при роботі з об’єктами.