ЛАБОРАТОРНА РОБОТА №3

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

Завдання:
Варіант 10. Створити клас – ціле число. У закритій частині визначити поля – система числення і рядок символів, що відповідає числу. Визначити необхідні конструктори, методи доступу, деструктор. Перевантажити потокові операції введення і виведення, вважаючи що слід вводити десяткові числа і систему числення, а виводити – число у обраній системі.

Код програми:

Furniture.h:

```
#pragma once
#include <iostream>

class Furniture {
private:
    char* room;
    int weight;
public:
    Furniture();
    Furniture(char* room, int weight);
    Furniture(const Furniture&);
    void setRoom(char* room);
    char* getRoom() const;
    void setWeight(int weight);
    int getWeight() const;
    void print() const;
    void input();
    ~Furniture();
    friend std::istream& operator >> (std::istream& is, Furniture& furn);
    friend std::ostream& operator <<(std::ostream& out, const Furniture& furn);

};
```

Furniture.cpp:

```
#include "Furniture.h"
#include <cassert>
#define N 32
using namespace std;

Furniture::Furniture() {
    cout << "Basic furniture constructor" << endl;
    this->weight = 0;
    room = nullptr;
}
Furniture::Furniture(char* room, int weight) {
    cout << "Parametrized furniture constructor" << endl;
    setRoom(room);
    setWeight(weight);
```

```cpp
}
Furniture::Furniture(const Furniture& src) {
    cout << "Parametrized furniture constructor (link)" << endl;
    setRoom(src.getRoom());
    setWeight(src.getWeight());
}
Furniture::~Furniture() {
    cout << "Furniture dectructor" << endl;
    if (room) {
        delete room;
    }
}

void Furniture::setRoom(char* room) {
    if (this->room)
        delete[] this->room;
    int roomLen = strlen(room) + 1;
    this->room = new char[roomLen];
    strcpy_s(this->room, roomLen, room);
}

void Furniture::setWeight(int weight) {
    this->weight = weight;
}

char* Furniture::getRoom() const {
    return this->room;
}

int Furniture::getWeight() const {
    return this->weight;
}

void Furniture::input() {
    char nroom[N];
    int nweight;
    cout << "Enter room: ";
    cin >> nroom;
    setRoom(nroom);
    cout << "Enter weight of furniture: ";
    cin >> nweight;
    setWeight(nweight);
}

void Furniture::print() const {
    cout << "The furniture for room \"" << getRoom() << "\" has weight " <<
getWeight() << endl;
}


istream& operator >> (istream& is, Furniture& furn ){
    char c, buff[N];

    is >> furn.weight;
    cin >> c;
    assert(c == '_');
    cin >> buff;
    furn.setRoom(buff);
    return is;
}

ostream& operator <<(ostream& os, const Furniture& furn)
{
    os << furn.room << "(" << furn.weight << ")";
    return os;
```

```
}
```

Int.h:

```
#pragma once
#include <iostream>

class Int {
private:
	int numSys;
	char* number;
	char* convDecToNumSys(int n, int numSys);
	//char* reverseNumber(char*);
public:
	Int();
	//Int(char* newNum, int newNumSys);
	Int(int newNum, int newNumSys);
	Int(const Int&);
	//void setNum(char* num);

	void setNum(int);
	char* getNum() const;
	void setSys(int);
	int getSys() const;
	//void print() const;
	void input();
	~Int();
	Int& operator = (const Int&);
	friend std::istream& operator >> (std::istream&, Int& );
	friend std::ostream& operator << (std::ostream&, const Int&);
};
```

Int.cpp:

```
#define _CRT_SECURE_NO_WARNINGS
#include "Int.h"
#include <cassert>
#define N 16
using namespace std;



Int::Int() {
	cout << "Basic Int constructor" << endl;
	this->numSys = 0;
	this->number = nullptr;
}
Int::Int(int newNum, int newSys){
	cout << "Parametrized Int constructor" << endl;
	setSys(newSys);
	setNum(newNum);
}
Int::Int(const Int& src) {
	cout << "Parametrized Int constructor (link)" << endl;
	int len = strlen(src.getNum()) + 1;
	this->number = new char[len];
	strcpy_s(this->number, len, src.getNum());

	setSys(src.getSys());
}

char* Int::convDecToNumSys(int n, int numSys)      //
{
```

```cpp
	// unsafe function
	char chNum[32];

	int i = 0;
	if (!(numSys == 2 || numSys == 8 || numSys == 10 || numSys == 16)) {
		cout << "wrong numerical system" << endl;
		exit(2);
		//return nullptr;
					// returns NULL if wrong numerical system
	}
	while (n != 0) {
		int temp = 0;
		temp = n % numSys;

		// according to ASCII table
		if (temp < 10) {
			chNum[i] = temp + 48;		// ASCII character of digits
		}
		else {
			chNum[i] = temp + 55;		// ASCII character of big letters
		}
		i++;
		n = n / numSys;
	}
	char* resNum = new char[++i];

	int k, j;
	for (k = 0, j = i - 2; j >= 0; k++, j--) {		// reversing string to
readable style
		resNum[k] = chNum[j];
	}
	resNum[k] = '\0';

	return resNum;
}
void Int::setNum(int n) {
	if (this->number) {
		delete[] this->number;
	}
	this->number = convDecToNumSys(n, this->numSys);
}
char* Int::getNum() const {

	return this->number;
}
void Int::setSys(int newSys) {
	this->numSys = newSys;
}
int Int::getSys() const {
	return this->numSys;
}

Int::~Int() {
	cout << "Int dectructor" << endl;
	if (this->number) {
		delete [] number;
	}
}

void Int::input() {
	cout << "Enter numerical system: ";
	cin >> numSys;
	int num;
	cout << "Enter a decimal nubmer: ";
	cin >> num;
	setNum(num);
}

istream& operator >> (istream& is, Int& obj) {
	// Enter a number using format: numerical_system:number_in_decimal_system
	char c;
	int num;
	is >> obj.numSys;
	cin >> c;
	assert(c == ':');
	cin >> num;
	obj.setNum(num);
	return is;
}
ostream& operator <<(ostream& out, const Int& obj) {
	out << obj.numSys << ":" << obj.number;
	return out;
}


Int& Int::operator = (const Int& obj) {
	this->numSys = obj.numSys;
	if (this->number) {
		delete[] this->number;
	}
	if (!obj.number)
		this->number = nullptr;					// it
would crash if read char* with nullptr
	else {
		int numLen = strlen(obj.number) + 1;
		this->number = new char[numLen];
		strcpy_s(this->number, numLen, obj.number);
	}
	return *this;
}
```

Integer.h:

```
#pragma once
#include <cstdint>
#include <iostream>

class Integer32 {
private:
	long int data;
public:
	Integer32();
	Integer32(long int);
	void setData(long int);
	int getData() const;
	void annul();
	Integer32 operator +(const Integer32& y) const;
	Integer32 operator -(const Integer32& y) const;
	Integer32 operator *(const Integer32& y) const;
	Integer32 operator /(const Integer32& y) const;
};
```

Integer.cpp:

```
#include "Integer.h"
#include <climits>
```

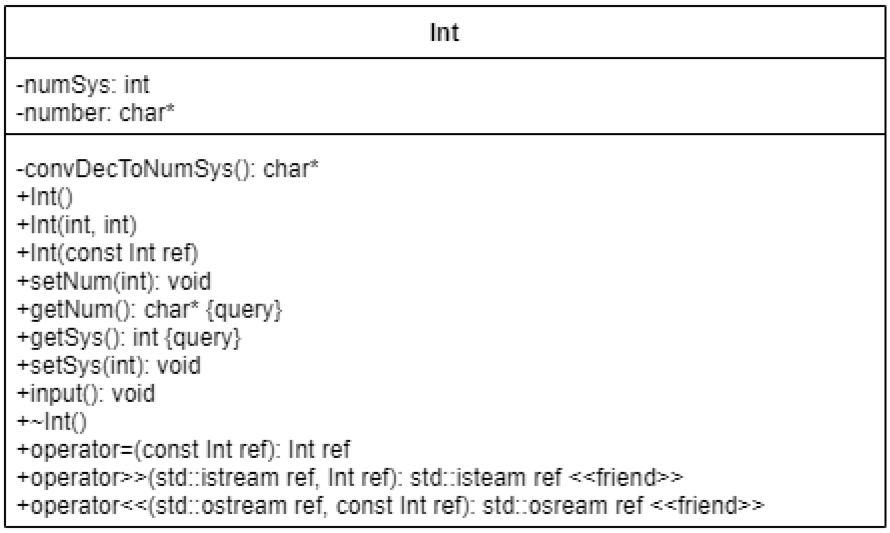
```
#include <stdlib.h>
#include <cmath>
#include <cstdint>
using namespace std;



Integer32::Integer32() {
	this->annul();
}

Integer32::Integer32(long int newData) {
	this->setData(newData);
}

int Integer32::getData() const{
	return this->data;
}
void Integer32::annul() {
	this->data = 0;
}

void Integer32::setData(long int newData) {
	this->data = newData;
}


Integer32 Integer32::operator +(const Integer32& y) const {
	long long int res = this->data;
	res += y.data;
	if (res > INT_MAX || res < INT_MIN) {
		cout << "overflow" << endl;
		exit(EXIT_FAILURE);
	}
	return Integer32(res);
}

Integer32 Integer32::operator -(const Integer32& y) const {
	return Integer32(this->data - y.data);
}

Integer32 Integer32::operator *(const Integer32& y) const {
	return Integer32(this->data * y.data);
}

Integer32 Integer32::operator /(const Integer32& y) const {
	if (!y.data) {
		cout << "Error: division by zero" << endl;
		exit(1);
	}
	return Integer32(this->data / y.data);
}
```

lab3.cpp:

```
#include <iostream>
#include "Integer.h"
#include "Furniture.h"
#include "string.h"
#include "Int.h"

#define N 16
using namespace std;
```

```
int getNumLen(int decNum) {
        int len = 0;
        while (decNum) {
                decNum /= 10;
                len++;
        }
        return len;
}
int main()
{
         //task 1
        Integer32 x(LONG_MIN), y, z(20);
        y = x + z;
        cout << y.getData() << endl;
        Integer32 f = x / z;
        cout << f.getData() << endl;

        // task 2
        char buff[N];
        strcpy_s(buff, N, "kitchen");
        Furniture furn1(buff, 42), furn2;
        cout << furn1 << endl;
        cout << "enter values in forman: [weight]_[room]" << endl;
        cin >> furn2;
        cout << furn2 << endl;

        // task 3
        Int one;
        one.setSys(8);
        one.setNum(2524);
        cout << one.getNum() << endl;
        Int two(one);
        two.setSys(2);
        two.setNum(127);
        Int three(255, 16);
        cout << one << endl << two << endl << three << endl;
        cout << "Enter number in format: [numerical_system]:[decimal_number]" << endl;
        cin >> one;
        cout << one << endl;
        Int four;
        three = four = three;
        cout << one << endl << two << endl << three << endl << four << endl;
        return 0;
}
```

UML класу з індивідуального завдання:

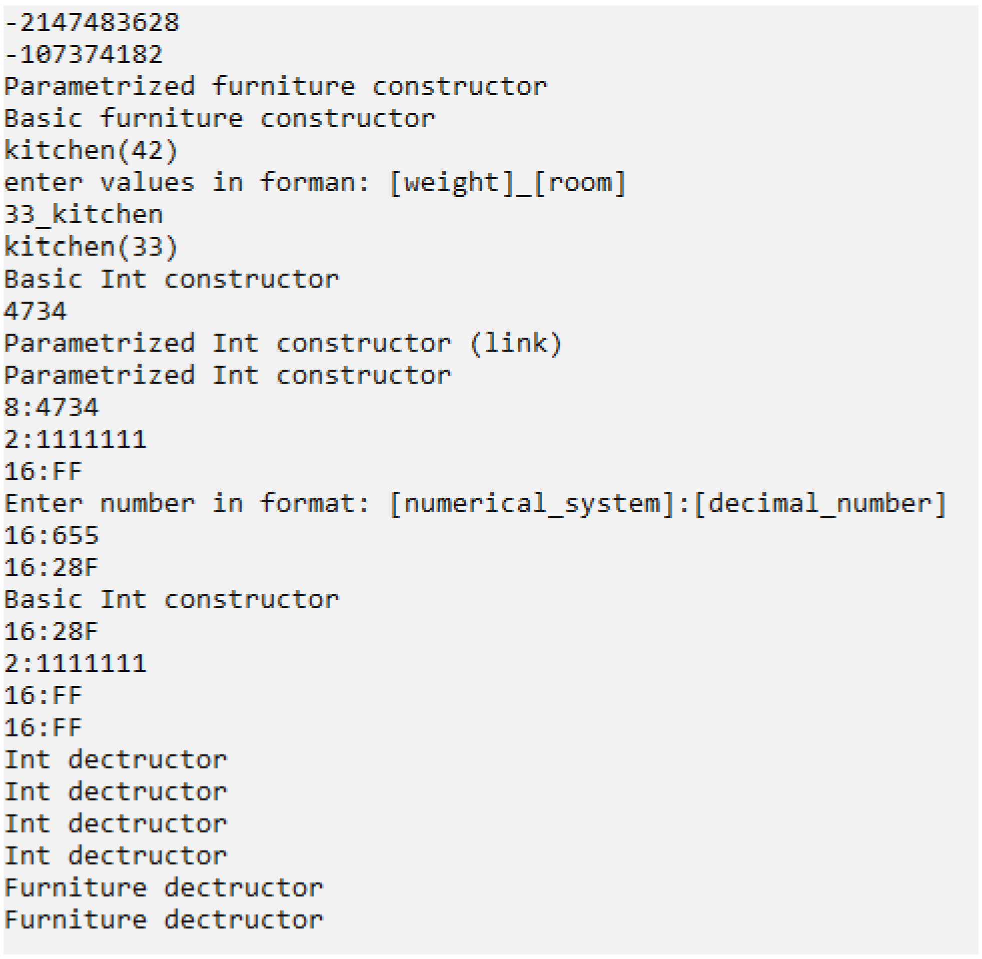


Int

-numSys: int
-number: char*

-convDecToNumSys(): char*
+Int()
+Int(int, int)
+Int(const Int ref)
+setNum(int): void
+getNum(): char* {query}
+getSys(): int {query}
+setSys(int): void
+input(): void
+~Int()
+operator=(const Int ref): Int ref
+operator>>(std::istream ref, Int ref): std::isteam ref <<friend>>
+operator<<(std::ostream ref, const Int ref): std::osream ref <<friend>>

Результат:



```
-2147483628
-107374182
Parametrized furniture constructor
Basic furniture constructor
kitchen(42)
enter values in forman: [weight]_[room]
33_kitchen
kitchen(33)
Basic Int constructor
4734
Parametrized Int constructor (link)
Parametrized Int constructor
8:4734
2:1111111
16:FF
Enter number in format: [numerical_system]:[decimal_number]
16:655
16:28F
Basic Int constructor
16:28F
2:1111111
16:FF
16:FF
Int dectructor
Int dectructor
Int dectructor
Int dectructor
Furniture dectructor
Furniture dectructor
```

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