Додаток 1

Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 4

з дисципліни «Основи програмування – 2. Метидології програмування»

«Перевантаження операторів»

Варіант 18

Виконав студент ІП-13 Король Валентин Олегович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 4

Варіант 18

Визначити клас "Numeral_8", членом якого є вісімкове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у двійкове, у тому числі і скороченим способом.
Перевантажити оператори: префіксний "++" - для інкрементації вісімкового числа, "+=" - для збільшення його на вказану величину, "+" - для додавання двох вісімкових чисел. Створити три вісімкових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у двійковий формат двома способами (звичайним і скороченим)

Код програми

C++

main.cpp

```
#include "Functions.h"
using namespace std;

int main()
{
	long long n1;
	cout << "Enter an octal number N1: "; cin >> n1;
	while (!isNumeral8(n1)) {
		cout << "You have entered a number that does not correspond to the octal
midrange! Enter N1 again: "; cin >> n1;
	}
	Numeral_8 N1(n1), N2(N1), N3;
	cout << "\nCurrent data:";
	outputNums(N1, N2, N3);

	long long n;
	cout << "\nEnter the amount by which you want to increase N2 (octal notation): "; cin
>> n;
	while (!isNumeral8(n)) {
```

```
            cout << "You entered a number in the wrong number system! Please try again: ";
cin >> n;
        }
        ++N1;
        N2 += n;
        N3 = N1 + N2;
        cout << "\nChanged data:";
        outputNums(N1, N2, N3);
        cout << "\nN3, converted to binary midrange in the usual way:  " <<
N3.convertToBinUsual();
        cout << "\nN3, converted to binary midrange in an abbreviated manner: " <<
N3.convertToBinReduced() << "\n\n";
        system("pause");
        return 0;
    }
```

c.h

```
#pragma once
#include <iostream>
#include <iomanip>
#include <string>
#include <stdio.h>
#include <math.h>
#include "Classes.h"
using namespace std;

void outputNums(Numeral_8, Numeral_8, Numeral_8);
bool isNumeral8(long long);
long long convertToDec(long long, int);
    long long convertFromDec(long long, int);
```

Functions.cpp

```
#include "Functions.h"
using namespace std;

void outputNums(Numeral_8 N1, Numeral_8 N2, Numeral_8 N3)
{
        cout << "\n       N1 = " << N1.getNum()
             << "\n       N2 = " << N2.getNum()
             << "\n       N3 = " << N3.getNum() << "\n";
}

bool isNumeral8(long long num)
{
        while (num) {
                if (num % 10 < 8) {
                        num /= 10;
                }
                else {
                        return false;
                }
        }
        return true;
}

long long convertToDec(long long num, int base)
{
        long long dec = 0;
        int i = 0;
        while (num) {
                dec += (num % 10) * (long long)pow(base, i++);
                num /= 10;
        }
        return dec;
```

```
}

long long convertFromDec(long long dec, int base)
{
	long long num = 0;
	int i = 0;
	while (dec > base - 1) {
		num += (dec % base) * (long long)pow(10, i++);
		dec /= base;
	}
	return num + dec * (long long)pow(10, i);
}
```

Classes.h

```
#pragma once
class Numeral_8
{
	long long number;
public:
	Numeral_8();
	Numeral_8(long long number);
	Numeral_8(Numeral_8& obj);

	long long getNum();
	long long convertToBinUsual();
	long long convertToBinReduced();
	Numeral_8& operator++();
	Numeral_8& operator+=(long long);
	Numeral_8 operator+(Numeral_8);
};
```

Classes.cpp

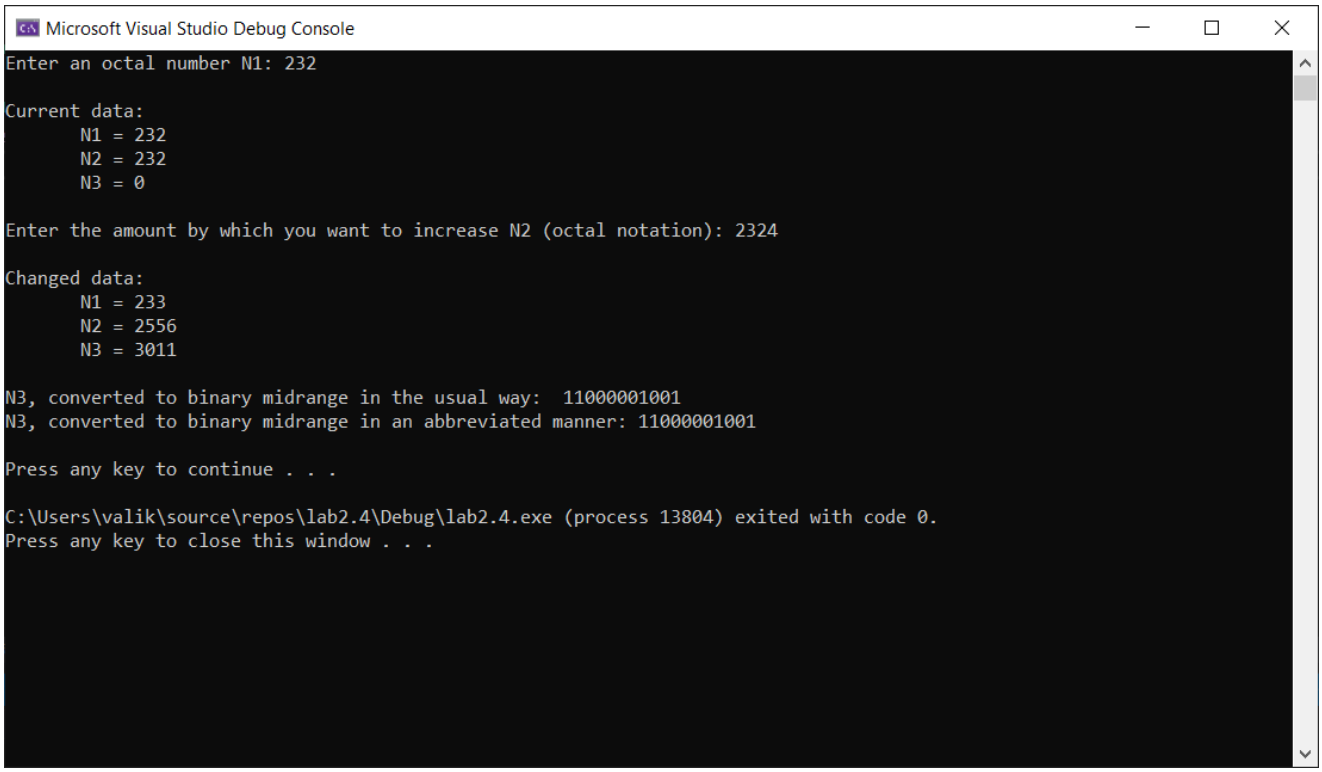
```
#include "Functions.h"
using namespace std;

Numeral_8::Numeral_8()
{
	this->number = 0;
}
Numeral_8::Numeral_8(long long number)
{
	this->number = number;
}
Numeral_8::Numeral_8(Numeral_8& obj)
{
	this->number = obj.number;
}

long long Numeral_8::getNum()
{
	return this->number;
}
long long Numeral_8::convertToBinUsual()
{
	long long dec = convertToDec(this->number, 8);
	long long bin = convertFromDec(dec, 2);
	return bin;
}
long long Numeral_8::convertToBinReduced()
{
```

```
	long long oct = this->number, bin = 0;
	int i = 0;
	while (oct) {
		bin += convertFromDec(oct % 10, 2) * (long long)pow(1000, i++);
		oct /= 10;
	}
	return bin;
}


Numeral_8& Numeral_8::operator++()
{
	this->number = convertFromDec(convertToDec(this->number, 8) + 1, 8);
	return *this;
}
Numeral_8& Numeral_8::operator+=(long long n)
{
	this->number = convertFromDec(convertToDec(this->number, 8) + convertToDec(n, 8), 8);
	return *this;
}
Numeral_8 Numeral_8::operator+(Numeral_8 obj)
{
	Numeral_8 sum(convertFromDec(convertToDec(this->number, 8) + convertToDec(obj.number,
8), 8));
	return sum;
}
```



Висновки:

Я вивчив та використав на практиці механізми створення класів з використання перевантаження операторів(операцій).