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.
Методології програмування.»

«Перевантаження операторів»

Варіант 28

Виконав студент ІП-11 Сідак Кирил Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №4

Мета: вивчити механізм створення класів з використанням перевантажених операторів(операцій).

Варіант 28

Визначити клас “Numeral_16”, членом якого є шістнадцяткове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у двійкове, у тому числі і скороченим способом. Перевантажити оператори: префіксний “++” - для інкрементації шістнадцяткового числа, “+=” - для збільшення його на вказану величину, “+” - для додавання двох шістнадцяткових чисел. Створити три шістнадцяткових 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).

Постановка задачі:

За умовою задачі треба визначити клас “Numeral_16”, атрибутом якого є шістнадцяткове число (тип цього атрибуту - рядок). Також потрібно реалізувати декілька конструкторів (за замовчуванням, з десятковим числом в якості параметра та з шістнадцятковим числом (рядок) в якості параметра) та геттер, який повертає рядкове представлення шістнадцяткового числа (рядковий атрибут). Крім того, треба реалізувати два методи перетворення шістнадцяткового числа у двійкове та перевантажити такі оператори: префіксний “++” - для інкрементації шістнадцяткового числа, “++” - для збільшення його на вказану величину, “+” - для додавання двох шістнадцяткових чисел. Використовуючи три різні конструктори, потрібно створити три шістнадцяткових числа N1, N2, N3, інкрементувати N1, збільшити на введену величину N2, а суму цих двох чисел зберегти в N3. Далі треба перевести отримане значення N3 у двійковий формат двома способами (звичайним і скороченим).

Програма на C++:

main.cpp

```
#include <iostream>
#include "Numeral_16.h"
#include "input_validation.h"

int main() {
    string choice;
    int n1_decimal;
    string n2_hex;
    cout << "Enter first hex number N1 in decimal format: ";
    cin >> n1_decimal;
    cin.ignore();
    cout << "Enter second hex number N2 in hex format:" <<
```

```
endl;
    getline(cin, n2_hex);
    Numeral_16 N1(n1_decimal), N2(n2_hex), N3;
    cout << "First hex number N1: " << N1.get_num_16_str() <<
endl;
    cout << "Second hex number N2: " << N2.get_num_16_str()
<< endl;
    cout << "N1 incremented: " << (++N1).get_num_16_str() <<
endl;
    choice = input_choice();
    if (choice == "d") {
        int num;
        cout << "Enter a number (in decimal format) to
increase N2 by: ";
        cin >> num;
        N2 += Numeral_16(num);
        cout << "N2 after adding " << num << " to it: " <<
N2.get_num_16_str() << endl;
    }
    else {
        string num;
        cout << "Enter a number (in hex format) to increase
N2 by:" << endl;
        getline(cin, num);
        N2 += Numeral_16(num);
        cout << "N2 after adding " << num << " to it: " <<
N2.get_num_16_str() << endl;
    }
    N3 = N1 + N2;
    cout << "Third hex number N3 (N3 = N1 + N2): " <<
N3.get_num_16_str() << endl;
    string N3_binary_1 = N3.convert_to_binary_long();
    cout << "N3 converted to binary using the first method: "
<< N3_binary_1 << endl;
    string N3_binary_2 = N3.convert_to_binary_short();
    cout << "N3 converted to binary using the second method:
" << N3_binary_2 << endl;
    return 0;
}
```

Numeral_16.h

```
#ifndef LAB_4_NUMERAL_16_H
#define LAB_4_NUMERAL_16_H
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
class Numeral_16 {
    string num_16_str;
public:
    Numeral_16() = default;
```

```
    explicit Numeral_16(const string& num_16_str) { this ->
num_16_str = num_16_str; }
    explicit Numeral_16(int);
    string get_num_16_str() const { return  num_16_str; }
    Numeral_16 operator ++();
    const Numeral_16 operator +(const Numeral_16& obj);
    const Numeral_16 operator += (const Numeral_16& obj);
    string convert_to_binary_short();
    string convert_to_binary_long();
};
#endif
```

Numeral_16.cpp

```
#include "Numeral_16.h"

Numeral_16::Numeral_16(int num_10) {
    if (!num_10) num_16_str = "0";
    else {
        int base = 16;
        string temp;
        int remainder;
        while (num_10) {
            remainder = num_10 % base;
            if (remainder >= 10) {
                temp += (char) (remainder + 55);
            }
            else {
                temp += (char) (remainder + 48);
            }
            num_10 /= base;
        }
        int length = temp.length();
        for (int j = length - 1; j >= 0; --j) {
            num_16_str += temp[j];
        }
    }
}

Numeral_16 Numeral_16::operator ++() {
    int num_10 = stoi(num_16_str, nullptr, 16);
    this -> num_16_str = Numeral_16(++num_10).num_16_str;
    return *this;
}

const Numeral_16 Numeral_16::operator +(const Numeral_16&
obj) {
    Numeral_16 temp;
    temp.num_16_str = Numeral_16(stoi(num_16_str, nullptr,
16) + stoi(obj.num_16_str, nullptr, 16)).num_16_str;
    return temp;
}
```

```
const Numeral_16 Numeral_16::operator +=(const Numeral_16&
obj) {
    num_16_str = Numeral_16(stoi(num_16_str, nullptr, 16) +
stoi(obj.num_16_str, nullptr, 16)).num_16_str;
    return *this;
}

string Numeral_16::convert_to_binary_short() {
    string num_2;
    for (char c : num_16_str) {
        switch (c) {
            case '0':
                num_2 += "0000";
                break;
            case '1':
                num_2 += "0001";
                break;
            case '2':
                num_2 += "0010";
                break;
            case '3':
                num_2 += "0011";
                break;
            case '4':
                num_2 += "0100";
                break;
            case '5':
                num_2 += "0101";
                break;
            case '6':
                num_2 += "0110";
                break;
            case '7':
                num_2 += "0111";
                break;
            case '8':
                num_2 += "1000";
                break;
            case '9':
                num_2 += "1001";
                break;
            case 'A':
                num_2 += "1010";
                break;
            case 'B':
                num_2 += "1011";
                break;
            case 'C':
                num_2 += "1100";
                break;
```

```
            case 'D':
                num_2 += "1101";
                break;
            case 'E':
                num_2 += "1110";
                break;
            case 'F':
                num_2 += "1111";
                break;
        }
    }
    int i = 0;
    bool found = true;
    string extra_zeros;
    while (i <= 3 && found) {
        found = false;
        if (num_2[i] == '0') {
            extra_zeros += num_2[i];
            found = true;
        }
        i++;
    }
    num_2.replace(0, extra_zeros.length(), "");
    return num_2;
}

string Numeral_16::convert_to_binary_long() {
    string num_2;
    int num_10 = stoi(num_16_str, nullptr, 16);
    if (!num_10) num_2 = "0";
    else {
        int base = 2;
        string temp;
        int remainder;
        while (num_10) {
            remainder = num_10 % base;
            temp += (char) (remainder + 48);
            num_10 /= base;
        }
        int length = temp.length();
        for (int j = length - 1; j >= 0; --j) {
            num_2 += temp[j];
        }
    }
    return num_2;
}
```

input_validation.h

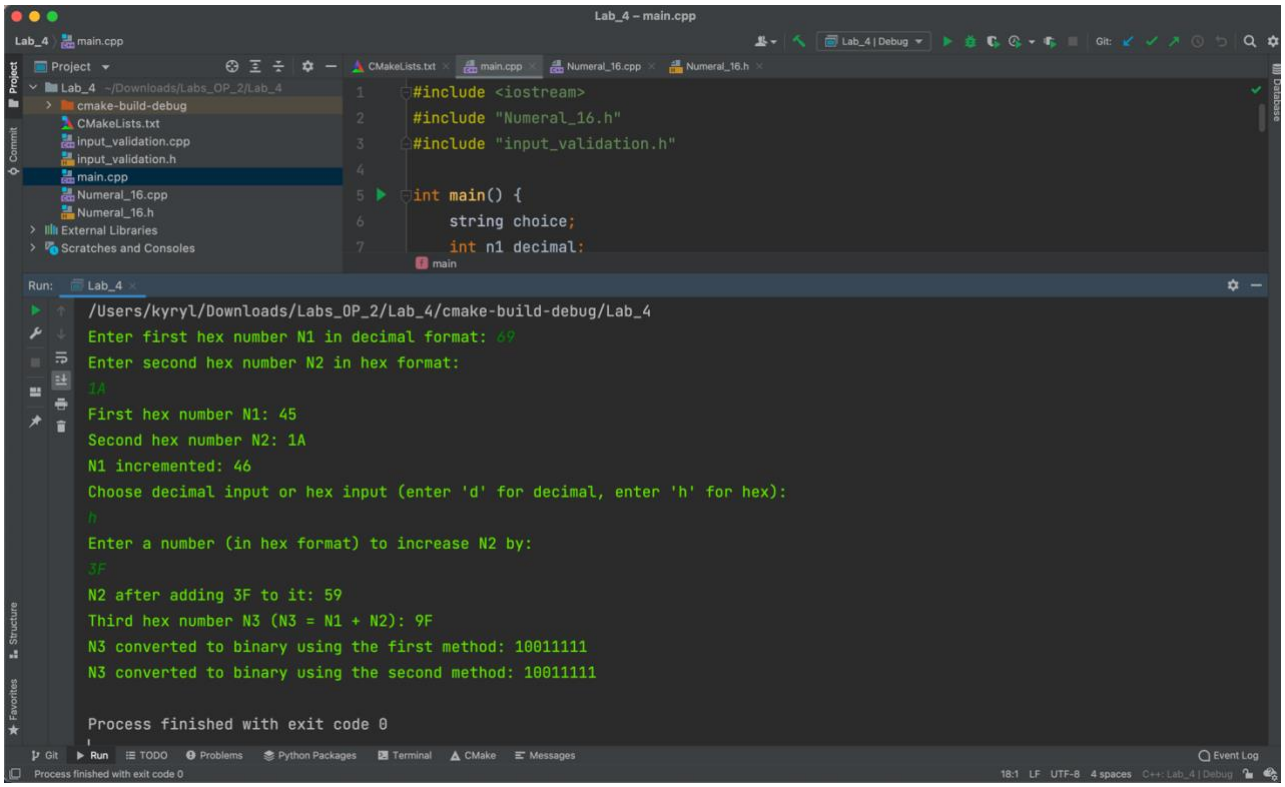
```
#ifndef LAB_4_INPUT_VALIDATION_H
#define LAB_4_INPUT_VALIDATION_H
#include <iostream>
```

```
using namespace std;

string input_choice();
#endif
```

input_validation.cpp

```
#include "input_validation.h"

string input_choice() {
    string choice;
    cout << "Choose decimal input or hex input (enter 'd' for
decimal, enter 'h' for hex):" << endl;
    getline(cin, choice);
    while (choice!= "d" and choice != "h") {
        cout << "Incorrect input. Enter 'd' or 'h'." << endl;
        cout << "Choose decimal input or hex input (enter 'd'
for decimal, enter 'h' for hex):" << endl;
        getline(cin, choice);
    }
    return choice;
}
```

Результат на C++:



Висновок

Отже, вивчити механізм створення класів з використанням перевантажених операторів(операцій) та застосував його на практиці, створивши клас, членом якого є шістнадцяткове число та перевантаживши для нього унарний оператор ++ для інкрементації шістнадцяткового числа та бінарні оператори += для збільшення шістнадцяткового числа на вказану величину і + для додавання двох шістнадцяткових чисел. Створивши три об’єкти цього класу та застосувавши ці перевантажені оператори для них, я отримав коректний результат.