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 № 3 з дисципліни

«Основи програмування - 2.

Модульне програмування.»

“Перевантаження операторів”

Варіант: 30

Виконав студент ІП-11 Тихонов Федір Сергійович

(шифр, прізвище, ім'я, по батькові)

Перевірив

(прізвище, ім'я, по батькові)

**Лабораторна робота №3:**

**“Перевантаження оперторів”**

**Мета:**  вивчити механізми створення класів з використанням перевантажених операторів.

**Постановка задачі:** Визначити клас “Roman\_numerals”, який задає число в римській формі запису(рядка). Реалізувати для цього класу класу декілька конструкторів, геттери, методи перетворення числа у десяткове. Перевантажити оператори: префіксний “++” для інкрементації даного числа, “+=” - для збільшення його на вказану величину(римське число), “+” - для додавання двох римських чисел. Створити три римських числа(R1, R2, R3), використавши різні конструктори. Інкрементувати число R1, а число R2 збільшити на вказану величину. Знайти суму змінених чисел R1, R2 та зберегти їх в R3. Перевести отримане значення N3 у десятковий формат.

**main.cpp**

**#include "lib.h"**

**using std::cout;**

**using std::endl;**

**int main(){**

**Roman\_numerals R1;**

**cout << "Initialising R1 with a default constructor: ";**

**R1.outputRoman();**

**Roman\_numerals R2("MCXIII");**

**cout << "Initialising R2 with a parameterized constructor: ";**

**R2.outputRoman();**

**Roman\_numerals R3(R2);**

**cout << "Initialising R3 with a copy constructor: ";**

**R3.outputRoman();**

**cout << "Incrementing a number R1(I):";**

**++R1;**

**R1.outputRoman();**

**int k = 121;**

**cout << "Adding a value" << k << " to an already existing number R2 :";**

**R2 += intToRoman(k);**

**R2.outputRoman();**

**cout << "Adding R2 and R1 and saving it in R3:" << endl;**

**R3 = R1 + R2;**

**R3.outputRoman();**

**cout << "Converting R3 to N3:" << endl;**

**int N3 = R3.convertToInt();**

**cout << "R3: ";**

**R3.outputRoman();**

**cout << "N3: " << N3 << endl;**

**return 0;**

**}**

**lib.cpp**

**#include "lib.h"**

**#include <utility>**

**using std::cout;**

**using std::string;**

**using std::vector;**

**using std::endl;**

**Roman\_numerals Roman\_numerals::operator+(Roman\_numerals rNum) {**

**int ArabNum1 = this->convertToInt();**

**int ArabNum2 = rNum.convertToInt();**

**int result = ArabNum1 + ArabNum2;**

**this->setRoman(intToRoman(result));**

**return \*this;**

**}**

**void Roman\_numerals::operator++() {**

**int ArabNum1 = this->convertToInt();**

**++ArabNum1;**

**this->setRoman(intToRoman(ArabNum1));**

**}**

**Roman\_numerals& Roman\_numerals::operator+=(Roman\_numerals rNum) {**

**int num1 = this->convertToInt();**

**int num2 = rNum.convertToInt();**

**num1+=num2;**

**setRoman(intToRoman(num1));**

**return \*this;**

**}**

**int Roman\_numerals::convertToInt() {**

**int answer = 0;**

**string copyNumeral = numeral;**

**int i = copyNumeral.length() - 1;**

**std::reverse(copyNumeral.begin(), copyNumeral.end());**

**while(copyNumeral[i] == 'M'){**

**answer += 1000;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**if (copyNumeral[i] == 'C' && copyNumeral[i - 1] == 'M') {**

**answer += 900;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i -= 2;**

**}**

**if (copyNumeral[i] == 'D') {**

**answer += 500;**

**copyNumeral.pop\_back();**

**i--;**

**while(copyNumeral[i] == 'C'){**

**answer += 100;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**}**

**if (copyNumeral[i] == 'C' && copyNumeral[i - 1] == 'D') {**

**answer += 400;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i -= 2;**

**}**

**while(copyNumeral[i] == 'C'){**

**answer += 100;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**if(copyNumeral[i] == 'X' && copyNumeral[i-1] == 'C'){**

**answer += 90;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i-=2;**

**}**

**if(copyNumeral[i] == 'L'){**

**answer += 50;**

**copyNumeral.pop\_back();**

**i--;**

**while(copyNumeral[i] == 'X'){**

**answer += 10;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**}**

**if (copyNumeral[i] == 'X' && copyNumeral[i - 1] == 'L') {**

**answer += 40;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i -= 2;**

**}**

**while(copyNumeral[i] == 'X'){**

**answer += 10;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**if(copyNumeral[i] == 'I' && copyNumeral[i-1] == 'X'){**

**answer += 9;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i-=2;**

**}**

**if(copyNumeral[i] == 'V'){**

**answer += 5;**

**copyNumeral.pop\_back();**

**i--;**

**while(copyNumeral[i] == 'I'){**

**answer += 1;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**}**

**if (copyNumeral[i] == 'I' && copyNumeral[i - 1] == 'V') {**

**answer += 4;**

**copyNumeral.pop\_back();**

**copyNumeral.pop\_back();**

**i -= 2;**

**}**

**while(copyNumeral[i] == 'I'){**

**answer += 1;**

**copyNumeral.pop\_back();**

**i--;**

**}**

**return answer;**

**}**

**Roman\_numerals::Roman\_numerals(string num) {**

**numeral = num;**

**}**

**void Roman\_numerals::setRoman(string num) {**

**numeral = num;**

**}**

**void Roman\_numerals::outputRoman() {**

**cout << numeral << endl;**

**}**

**void Roman\_numerals::operator++(int) {**

**int ArabNum1 = this->convertToInt();**

**ArabNum1++;**

**this->setRoman(intToRoman(ArabNum1));**

**}**

**string Roman\_numerals::getRoman() {**

**return numeral;**

**}**

**Roman\_numerals::Roman\_numerals() {**

**numeral = "I";**

**}**

**Roman\_numerals::Roman\_numerals(const Roman\_numerals &toBeCopied) {**

**this->numeral = toBeCopied.numeral;**

**}**

**vector<int> intToVector(int num) {**

**string line;**

**int k;**

**vector<int> numerals = {};**

**string strNum = std::to\_string(num);**

**for (char i: strNum) {**

**line = "";**

**line.push\_back(i);**

**k = std::stoi(line);**

**numerals.push\_back(k);**

**}**

**return numerals;**

**}**

**string intToRoman(int num) {**

**vector<int> numbers = intToVector(num);**

**int len = numbers.size();**

**string RomanNumeral;**

**int i = 0;**

**int tempNum = numbers[i];**

**if (len > 3) {**

**for (int k = 0; k < tempNum; k++) {**

**RomanNumeral += 'M';**

**}**

**len--;**

**i++;**

**}**

**tempNum = numbers[i];**

**if (len > 2 && tempNum < 4) {**

**for (int k = 0; k < tempNum; k++) {**

**RomanNumeral.push\_back('C');**

**}**

**len--;**

**i++;**

**} else if (len > 2 && tempNum == 4) {**

**RomanNumeral += "CD";**

**len--;**

**i++;**

**} else if (len > 2 && tempNum == 5) {**

**RomanNumeral.push\_back('D');**

**len--;**

**i++;**

**} else if (len > 2 && tempNum > 5 && tempNum < 9) {**

**RomanNumeral.push\_back('D');**

**for (int k = 0; k < tempNum - 5; k++) {**

**RomanNumeral.push\_back('C');**

**}**

**len--;**

**i++;**

**} else if (len > 2 && tempNum == 9) {**

**RomanNumeral += "CM";**

**len--;**

**i++;**

**}**

**tempNum = numbers[i];**

**if (len > 1 && tempNum < 4) {**

**for (int k = 0; k < tempNum; k++) {**

**RomanNumeral.push\_back('X');**

**}**

**len--;**

**i++;**

**}**

**else if (len > 1 && tempNum == 4) {**

**RomanNumeral += "XL";**

**len--;**

**i++;**

**} else if (len > 1 && tempNum == 5) {**

**RomanNumeral += "L";**

**len--;**

**i++;**

**} else if (len > 1 && tempNum > 5 && tempNum < 9) {**

**RomanNumeral.push\_back('L');**

**for (int k = 0; k < tempNum - 5; k++) {**

**RomanNumeral.push\_back('X');**

**}**

**len--;**

**i++;**

**} else if (len > 1 && tempNum == 9) {**

**RomanNumeral += "XC";**

**len--;**

**i++;**

**}**

**tempNum = numbers[i];**

**if (len > 0 && tempNum < 4) {**

**for (int k = 0; k < tempNum; k++) {**

**RomanNumeral.push\_back('I');**

**}**

**} else if (len > 0 && tempNum == 4) {**

**RomanNumeral += "IV";**

**len--;**

**i++;**

**} else if (len > 0 && tempNum == 5) {**

**RomanNumeral.push\_back('V');**

**len--;**

**i++;**

**} else if (len > 0 && tempNum > 5 && tempNum < 9) {**

**RomanNumeral.push\_back('V');**

**for (int k = 0; k < tempNum - 5; k++) {**

**RomanNumeral.push\_back('I');**

**}**

**len--;**

**i++;**

**} else if (len > 0 && tempNum == 9) {**

**RomanNumeral += "IX";**

**len--;**

**i++;**

**}**

**return RomanNumeral;**

**}**

**lib.h**

**#include <iostream>**

**#include <vector>**

**#include <string>**

**using std::string;**

**using std::vector;**

**class Roman\_numerals{**

**public:**

**Roman\_numerals(string num);**

**Roman\_numerals();**

**Roman\_numerals(const Roman\_numerals& toBeCopied);**

**Roman\_numerals operator+(Roman\_numerals rNum);**

**Roman\_numerals& operator+=(Roman\_numerals rNum);**

**void operator++();**

**void operator++(int);**

**void outputRoman();**

**void setRoman(string num);**

**string getRoman();**

**int convertToInt();**

**private:**

**string numeral;**

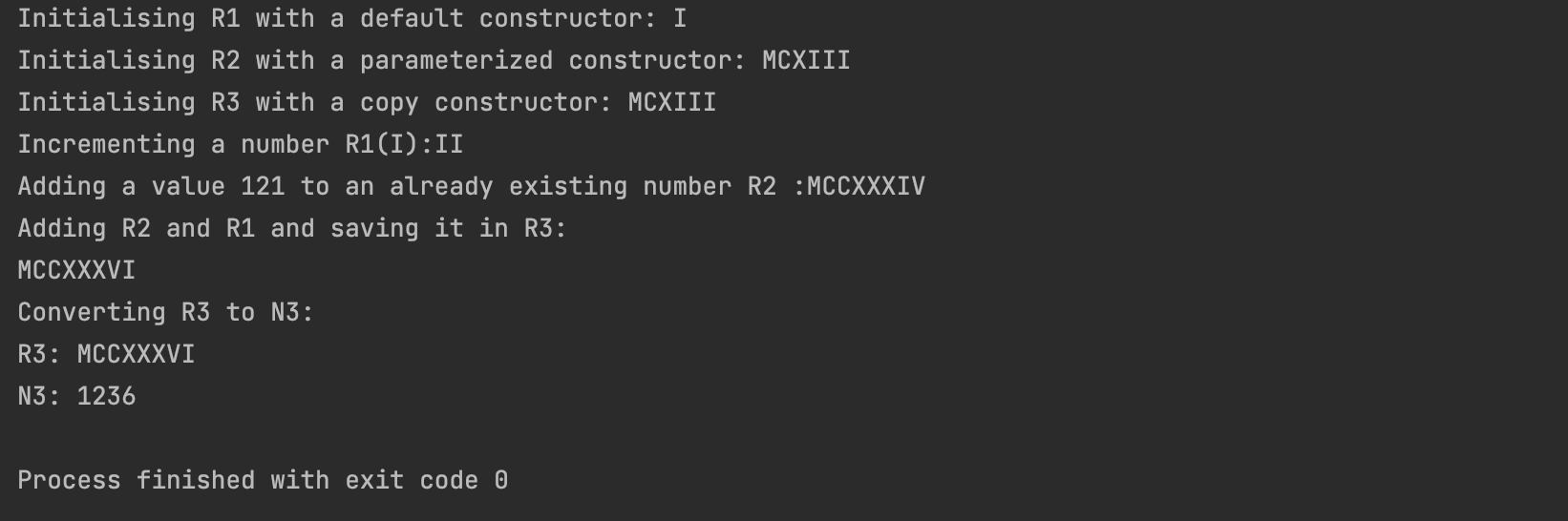
**};**

**vector<int> intToVector(int num);**

**string intToRoman(int num);**

**string getNumeral();**

**Результати на С++:**

****

**Висновок:**

**Отже, ми навчилися перевантажувати оператори, вивчили особливості роботи класів з перевантаженими операторами.**