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. Модульне програмування»
«Перевантаження операторів»
Варіант 7

Виконав студент ІП-15, Гуменюк Олександр Володимирович
(шифр, прізвище, ім'я, по батькові)
Перевірила Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 4

Перевантаження операторів

Індивідуальне завдання

Варіант 7

7. Визначити клас "Дата" для роботи із датами в межах року. Членами класу є число, місяць та рік. Реалізувати для нього декілька конструкторів, геттери, метод визначення пори року, що відповідає вказаній даті. Перевантажити оператори: "+=" – для збільшення дати на вказану кількість днів, "-" – для знаходження інтервалу між двома датами. Створити три об'єкта-дати (D1, D2, D3), використовуючи різні конструктори. Збільшити дату D1 на 9 днів, а дату D2 – на 14 днів. Визначити тривалість інтервалу між датами D1 і D2. Для дати D3 визначити пору року, якій відповідає ця дата.

Код C++

OP_Lab1.cpp

```cpp
#include "functions.h"

int main()
{
    Date d1, d2, d3;
    inputDates(d1, d2, d3);

    printDates(d1, d2, d3, "\nBefore Calculations");

    Date interval = performCalculations(d1, d2, d3);

    printDates(d1, d2, d3, "After calculations");
    cout << "The interval between D1 and D2 is "; interval.printInterval();
    cout << "The season of D3 is " << d3.getSeason() << endl;

}
```

functions.h

```cpp
#pragma once

#include <string>
#include <iostream>
#include "date.h"

using namespace std;

bool isNum(string s);
int inputNum(string text);
void inputDates(Date& d1, Date& d2, Date& d3);
void printDates(Date& d1, Date& d2, Date& d3, string header);
Date performCalculations(Date& d1, Date& d2, Date& d3);
```

functions.cpp

```cpp
#include "functions.h"

bool isNum(string s)
{
    for (char ch : s) {
        if (!isdigit(ch)) return false;
    }
    if (stoi(s) == 0) return false;

    return true;
}


int inputNum(string text) {
    string n;
    cout << text;
    cin >> n;

    while (!isNum(n)) {
        cout << "You can only enter a positive integer: ";
        cin >> n;
    }
    cin.ignore();
    return stoi(n);
}
```

```cpp
void inputDates(Date& d1, Date& d2, Date& d3) {

    cout << "Dates are entered and displayed in logical view: 1 <= year, 1 <= month <= 12, 1 <= day <= 28-31 \n"
            "However, for all dates (years, month, days) you may enter any positive integer \n"
            "The program will automatically reformat the date \n" << endl;

    int year, month, day;
    year = inputNum("Input year number for D1: ");
    month = inputNum("Input month number for D1: ");
    day = inputNum("Input day number for D1: ");
    d1 = Date(day, month, year);

    month = inputNum("\nInput month number for D2: ");
    day = inputNum("Input day number for D2: ");
    d2 = Date(day, month);

    day = inputNum("\nInput day number for D3: ");
    d3 = Date(day);
}

void printDates(Date& d1, Date& d2, Date& d3, string header) {
    cout << header << endl;
    cout << "D1: "; d1.printDate();
    cout << "D2: "; d2.printDate();
    cout << "D3: "; d3.printDate();
    cout << "\n";
}
```

```cpp
Date performCalculations(Date& d1, Date& d2, Date& d3) {
    d1 += 9;
    d2 += 14;
    Date interval = d1 - d2;
    return interval;
}
```

date.h

```h
#pragma once

#include <string>
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

class Date {
    int day, month, year;
    static const vector<int> monthsOfYear;
    static const vector<int> monthsOfLeapYear;

    void formatDate();
    void formatYears();
    void formatMonths();
    bool isLeapYear(int year) const;
    int toDays() const;
public:
    Date() :day(0), month(0), year(0) {};
    Date(int days, int months, int years);
    Date(int days, int months);
    Date(int days);

    const Date operator+=(const int days);
    const Date operator-(const Date date);
    int getDay();
    int getMonth();
    int getYear();
    string getSeason();
    void printDate();
    void printInterval();
};
```

date.cpp

```cpp
#include "date.h"

const vector<int> Date::monthsOfYear =     { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
const vector<int> Date::monthsOfLeapYear = { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

Date::Date(int days, int months, int years) {
    this->day = days - 1;
    this->month = months - 1;
    this->year = years - 1;
    formatDate();
}

Date::Date(int days, int months) {
    this->day = days - 1;
    this->month = months - 1;
    this->year = 0;
    formatDate();
}

Date::Date(int days) {
    this->day = days - 1;
    this->month = 0;
    this->year = 0;
    formatDate();
}


bool Date::isLeapYear(int year) const {
    int tempYear = year + 1;
    return ((tempYear % 4 == 0 and tempYear % 100 != 0) or tempYear % 400 == 0);
}
```

```cpp
void Date::formatYears() {
    year += month / 12;
    month = month % 12;

    while (day >= 366 || (day >= 365 && !isLeapYear(year))) {

        if (isLeapYear(year)) {
            day -= 366;
            year += 1;
        }
        else {
            day -= 365;
            year += 1;
        }
    }
}

void Date::formatMonths() {
    vector<int> currentMonthsOfYear;

    currentMonthsOfYear = isLeapYear(year) ? monthsOfLeapYear : monthsOfYear;

    while (day >= monthsOfYear[month]) {
        day -= monthsOfYear[month];
        month += 1;
        if (month == 12) {
            currentMonthsOfYear = isLeapYear(year) ? monthsOfLeapYear : monthsOfYear;
            year += 1;
            month = 0;
        }
    }
}
```

```cpp
void Date::formatDate() {
    formatYears();
    formatMonths();
}

const Date Date::operator+=(const int days) { ... }

int Date::toDays() const {
    int days = day;
    int years = year;

    while (years >= 0) {
        if (isLeapYear(years)) days += 366;
        else                   days += 365;
        years -= 1;
    }

    int months = month;

    vector<int> currentMonthsOfYear = isLeapYear(year) ? monthsOfLeapYear : monthsOfYear;

    while (months >= 0) {
        days += currentMonthsOfYear[months];
        months--;
    }

    return days + 1;
}
```

```cpp
const Date Date::operator-(const Date date) {
    int days1 = this->toDays();
    int days2 = date.toDays();

    Date interval(abs(days1 - days2));

    return interval;
}

string Date::getSeason() {
    if (month < 2 || month == 11)  return "Winter";
    else if (month < 5)            return "Spring";
    else if (month < 8)            return "Summer";
    else                           return "Autumn";
}

int Date::getDay() {
    return day;
}
int Date::getMonth() {
    return month;
}
int Date::getYear() {
    return year;
}

void Date::printDate() {
    cout << "Year: " << year + 1<< " Month: " << month +1 << " Day: " << day + 1 << endl;
}

void Date::printInterval() {
    cout << year << " years " << month << " months " << day << " days" << endl;
}
```

Тестування C++

```
C:\WINDOWS\system32\cmd.exe
Dates are entered and displayed in logical view: 1 <= year, 1 <= month <= 12, 1 <= day <= 28-31
However, for all dates (years, month, days) you may enter any positive integer
The program will automatically reformat the date

Input year number for D1: 2022
Input month number for D1: 10
Input day number for D1: 10

Input month number for D2: 2000
Input day number for D2: 10

Input day number for D3: 10000

Before Calculations
D1: Year: 2022 Month: 10 Day: 10
D2: Year: 167 Month: 8 Day: 10
D3: Year: 28 Month: 5 Day: 19

After calculations
D1: Year: 2022 Month: 10 Day: 19
D2: Year: 167 Month: 8 Day: 24
D3: Year: 28 Month: 5 Day: 19

The interval between D1 and D2 is 1855 years 1 months 25 days
The season of D3 is Spring
Press any key to continue . . .
```