МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**

**КУРСОВАЯ РАБОТА**

**по дисциплине: «***Программирование***»**

**на тему:** **«**дата (день, месяц, год).**»**

Выполнил(а):Проверил:

Студент гр. «АТ-33», « АВТФ» *«Воронов В. В.»*

*«Мельничук В. Д.» «\_\_\_»*

«21» декабря 2024г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2024

**Техническое задание:**

1. Разработка класса структуры данных: дата (день, месяц, год). Далее Date.

Класс должен включать в себя (если это необходимо и подходит по смыслу к ТД):

* Конструкторы (по умолчанию, с параметрами);
* Деструктор;
* Функцию вывода на экран;
* Перегруженные операторы:
  + Оператор присваивания;
  + Бинарные арифметические (+, -,) с операндами ТД и ТД, ТД и базовый тип, базовый тип и ТД;
  + Арифметика с накоплением (+=,-=);
  + Унарные (++,--) в префиксной и постфиксной форме;
  + Логические (<, >, ==, != );
  + Операторы преобразования типа к любому базовому.

Написать головную программу, обеспечивающую проверку всех реализованных свойств и методов класса.

**Описание разработанного класса Date:**

Класс Date хранит в себе день, месяц и год, содержит проверку валидности года (для конструктора с днем, месяцем и годом), 3 конструктора, деструктор, операции с различными операторами, функцию вывода на экран и преобразование в тип string.

**Функции:**

- Конструкторы – создают объект класса Date. Конструктор по умолчанию, задающий 1 января 2025 года, конструктор с параметрами, принимающий три параметра int(day, month, year) и конструктор копии.

- Деструктор – уничтожает объект класса Date, освобождая внутреннюю память.

**Перегруженные операторы:**

- operator= – присваивает значения Date.

**-** operator+ – прибавляет к дате несколько дней.

- operator+= – прибавляет к дате несколько дней.

- operator++ – прибавляет к дате один день(постфиксный и префиксный)

- operator== – сравнивает две даты

- operator!= – сравнивает две даты

- operator> – сравнивает две даты

- operator< – сравнивает две даты

- operator string() – преобразует дату к типу string

**Листинг:**

Date.cpp

#include <iostream>

#include <stdexcept>

using namespace std;

class Date {

private:

    int day;

    int month;

    int year;

    bool isValidDate(int d, int m, int y) {

        if (y < 0 || m < 1 || m > 12 || d < 1) return false;

        int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        if ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0)) daysInMonth[1] = 29; // Leap year

        return d <= daysInMonth[m - 1];

    }

public:

    Date() : day(1), month(1), year(2025) {}

    Date(int d, int m, int y) {

        if (!isValidDate(d, m, y)) throw invalid\_argument("Invalid date");

        day = d;

        month = m;

        year = y;

    }

    Date(const Date &other) : day(other.day), month(other.month), year(other.year) {}

    ~Date() {}

    Date &operator=(const Date &other) {

        if (this != &other) {

            day = other.day;

            month = other.month;

            year = other.year;

        }

        return \*this;

    }

    Date operator+(int days) const {

        Date result(\*this);

        while (days > 0) {

            int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

            if ((result.year % 4 == 0 && result.year % 100 != 0) || (result.year % 400 == 0)) {

                daysInMonth[1] = 29;

            }

            int remainingDays = daysInMonth[result.month - 1] - result.day;

            if (days <= remainingDays) {

                result.day += days;

                break;

            } else {

                days -= (remainingDays + 1);

                result.day = 1;

                result.month++;

                if (result.month > 12) {

                    result.month = 1;

                    result.year++;

                }

            }

        }

        return result;

    }

    Date &operator+=(int days) {

        \*this = \*this + days;

        return \*this;

    }

    Date &operator++() {

        \*this += 1;

        return \*this;

    }

    Date operator++(int) {

        Date temp(\*this);

        ++(\*this);

        return temp;

    }

    bool operator<(const Date &other) const {

        if (year != other.year) return year < other.year;

        if (month != other.month) return month < other.month;

        return day < other.day;

    }

    bool operator>(const Date &other) const {

        if (year != other.year) return year > other.year;

        if (month != other.month) return month > other.month;

        return day > other.day;

    }

    bool operator==(const Date &other) const {

        return day == other.day && month == other.month && year == other.year;

    }

    bool operator!=(const Date &other) const {

        return !(\*this == other);

    }

    friend ostream &operator<<(ostream &os, const Date &date) {

        os << (date.day < 10 ? "0" : "") << date.day << "/"

           << (date.month < 10 ? "0" : "") << date.month << "/" << date.year;

        return os;

    }

    operator string() const {

        return (day < 10 ? "0" : "") + to\_string(day) + "/" +

               (month < 10 ? "0" : "") + to\_string(month) + "/" + to\_string(year);

    }

};

main.cpp

#include "Date.cpp"

int main() {

    try {

        Date date1(31, 12, 2023);

        Date date2 = date1 + 1;

        cout << "Date1: " << date1 << endl;

        cout << "Date2: " << date2 << endl;

        date1 += 365;

        cout << "Date1 + 365: " << date1 << endl;

        cout << "date1 < date2: " << (date1 < date2) << endl;

        cout << "Date1++: " << date1++ << endl;

        cout << "Date1 as string: " << string(date1) << endl;

    } catch (const exception &e) {

        cerr << e.what() << endl;

    }

    return 0;

}