# Лабораторная работа №1. Классы и объекты в С++

## Задание

Определить класс «Дата», в котором производится ввод календарной даты, определение соответствующего дате дня недели, прибавления к дате целой константы и т.п. операции.

## Используемый алгоритм для вычисления дня недели по дате

Для начала за начальную дату, как точку отсчеты, была выбрана 1 января 1900 (понедельник). Далее вычислялось количество дней пройденных с начальной даты до даты, записанной в классе (данный алгоритм прост, для пояснения ниже описал, как вычислялся високосный год). От подсчитанного количества дней подсчитывался остаток от деления на 7 (число дней в неделю), и соответствующей цифре соответствовал день недели (0 – воскресенье, 1 – понедельник и т.д.).

Год является високосным в двух случаях: либо он кратен 4, но при этом не кратен 100, либо кратен 400, что соответствует результаты булевского выражения: ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0).

## Диаграмма класса

|  |
| --- |
| Date |
| -\_day :int (День)  -\_month : int (Месяц)  -\_year : int (Год)  - :enum  {  DAY\_MIN = 1, (Минимальный день)  MONTH\_MIN = 1, (Минимальный месяц)  MONTH\_MAX = 12, (Максимальный месяц)  YEAR\_MIN = 1900, (Минимальный год)  YEAR\_MAX = 2400 (Максимальный год)  } |
| +Date (void): (Коструктор)  +Date(int day, int month, int year): (Коструктор)  +~Date(void): (Деструктор)  +getDayOfTheWeek(): string (Получение дня недели по дате)  +Operator +=(int n): Date (Прибавление к дате числа, т.е. количества дней)  +Operator -=(int n): Date (Вычитания от да ты числа)  +Operator bool: bool (Приведение даты к типу bool для проверки верности даты)  -IsValidDate(int day, int month, int year): bool (Проверки верности даты)  -IsValidYear(int year) : bool(Проверки верности года)  -IsValidMonth(int month) : bool (Проверки верности месяца)  -IsValidDay(int day, int month, int year) : bool (Проверки верности дня)  -GetLengthOfMonth(int month, int year) : int (Метод для указания количества дней в месяце)  -IsLeapYear(int year) : bool (Проверки високосности года)  -AddDay(int lengthOfMonth = 0) : (Прибаление дня к дате)  -AddMonth() : (Прибаление месяца к дате)  -SubtractDay() : (Вычитания дня из дате)  -SubtractMonth() : (Вычитания месяца из дате)  -GetCountLeapYear() : int (Вычисление високосных годов между начальной датой и датой, находящейся в экземпляре класса)  -GetCountDays() : int (Вычисление количества дней между начальной датой и датой, находящейся в экземпляре класса)  -GetDayOfTheWeek(int days) : string (Преобразования вычисления в выдаваемый вид недели) |

## Перечень сигнатур методов с комментариями согласно заданию

Date.h:

#pragma once

#include <string>

using namespace std;

const string MESSAGE\_ERROR = "Incorrect date!";

class Date

{

private:

static enum

{

DAY\_MIN = 1,

MONTH\_MIN = 1,

MONTH\_MAX = 12,

YEAR\_MIN = 1900,

YEAR\_MAX = 2400,

};

int \_day;

int \_month;

int \_year;

bool IsValidDate(int day, int month, int year) const;

bool IsValidYear(int year) const;

bool IsValidMonth(int month) const;

bool IsValidDay(int day, int month, int year) const;

int GetLengthOfMonth(int month, int year) const;

bool IsLeapYear(int year) const;

void AddDay(int lengthOfMonth = 0);

void AddMonth();

void SubtractDay();

void SubtractMonth();

int GetCountLeapYear() const;

int GetCountDays() const;

string GetDayOfTheWeek(int days) const;

public:

Date(void);

Date(int day, int month, int year);

int GetDay() const;

int GetMonth() const;

int GetYear() const;

string GetDayOfTheWeek() const;

//Перегруженные операции +=, -=(для сложения или вычитания дней из даты)

//и приведение к типу bool(для проверки действительности даты)

operator bool() const;

Date& operator+= (int n);

Date& operator-= (int n);

friend istream& operator>>(istream& cin, Date& date);

friend ostream& operator<<(ostream& cout, const Date& date);

~Date(void);

## }

## Листинг

Date.cpp:

#include "Date.h"

//----------------------------------------Конструкторы и деструктор -------------------------------

Date::Date(void)

{

\_day = DAY\_MIN;

\_month = MONTH\_MIN;

\_year = YEAR\_MIN;

}

Date::Date(int day, int month, int year)

{

\_day = day;

\_month = month;

\_year = year;

}

Date::~Date(void)

{

}

//--------------------------------------------------------------------------------------------------

//----------------------------------- Все для проверки правильности даты --------------------------

bool Date::IsValidDate(int day, int mount, int year) const

{

return IsValidYear(year) && IsValidMonth(mount) && IsValidDay(day, mount, year);

}

bool Date::IsValidYear(int year) const

{

return year >= YEAR\_MIN && year <= YEAR\_MAX;

}

bool Date::IsValidMonth(int month) const

{

return month >= MONTH\_MIN && month <= MONTH\_MAX;

}

bool Date::IsValidDay(int day, int month, int year) const

{

return day >= DAY\_MIN && day <= GetLengthOfMonth(month, year);

}

int Date::GetLengthOfMonth(int month, int year) const

{

int length = 31;

switch (month)

{

case 2:

length = IsLeapYear(year) ? 29 : 28;

break;

case 4:

case 6:

case 9:

case 11:

length = 30;

break;

}

return length;

}

bool Date::IsLeapYear(int year) const

{

return ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0);

}

//---------------------------------------------------------------------------------------

//-------------------------------------- Доступ к значениям полей класса ----------------

int Date::GetDay() const

{

return \_day;

}

int Date::GetMonth()const

{

return \_month;

}

int Date::GetYear() const

{

return \_year;

}

//---------------------------------------------------------------------------------------

//------------------------------------------Операции с датой ----------------------------

void Date::AddDay(int lengthOfMonth)

{

if(\_day < (lengthOfMonth != 0 ? lengthOfMonth : this->GetLengthOfMonth(\_month, \_year)))

{

++\_day;

}

else

{

AddMonth();

\_day = 1;

}

}

void Date::AddMonth()

{

if(\_month < MONTH\_MAX)

{

++\_month;

}

else

{

++\_year;

\_month = 1;

}

}

void Date::SubtractDay()

{

if(\*this)

{

if(\_day > 1)

{

--\_day;

}

else

{

SubtractMonth();

\_day = GetLengthOfMonth(\_month, \_year);

}

}

}

void Date::SubtractMonth()

{

if(\_month > 1)

{

--\_month;

}

else

{

--\_year;

\_month = MONTH\_MAX;

}

}

//---------------------------------------------------------------------------------------

//--------------------------------------------Другие функции ----------------------------

string Date::GetDayOfTheWeek() const

{

string dayOfTheWeek;

if (\*this)

{

dayOfTheWeek = GetDayOfTheWeek(GetCountDays() % 7);

}

else

{

dayOfTheWeek = MESSAGE\_ERROR;

}

return dayOfTheWeek;

}

string Date::GetDayOfTheWeek(int days) const

{

string day;

switch (days)

{

case 0:

day = "Sunday";

break;

case 1:

day = "Monday";

break;

case 2:

day = "Tuesday";

break;

case 3:

day = "Wednesday";

break;

case 4:

day = "Thursday";

break;

case 5:

day = "Friday";

break;

case 6:

day = "Saturday";

break;

}

return day;

}

int Date::GetCountDays() const

{

int days = (\_year - YEAR\_MIN) \* 365 + GetCountLeapYear() - (IsLeapYear(\_year) ? 1 : 0);

for (int i = 1; i < \_month; i++)

{

days += GetLengthOfMonth(i, \_year);

}

days += \_day;

return days;

}

int Date::GetCountLeapYear() const

{

int different = \_year - YEAR\_MIN;

int count = (int)(different / 4) - (int)(different / 100) + (int)(\_year / 400 - YEAR\_MIN / 400);

return count;

}

//---------------------------------------------------------------------------------------

//-----------------------------------------Перегруженные операции -----------------------

Date::operator bool() const

{

return IsValidDate(\_day, \_month, \_year);

}

Date& Date::operator+= (int n)

{

if(\*this)

{

int length = GetLengthOfMonth(\_month, \_year);

for (int i = 0; i < n; i++)

{

AddDay(length);

}

}

return \*this;

}

Date& Date::operator-= (int n)

{

if(\*this)

{

for (int i = 0; i < n; i++)

{

SubtractDay();

}

}

return \*this;

}

//---------------------------------------------------------------------------------------

//----------------------------Перегрузка операторов ввода/вывода ------------------------

istream& operator>>(istream& cin, Date& date)

{

return cin >> date.\_day >> date.\_month >> date.\_year;

}

ostream& operator<<(ostream& cout, const Date& date)

{

if(date)

{

cout << date.\_day << "." << date.\_month << "." << date.\_year ;

}

else

{

cout << MESSAGE\_ERROR;

}

return cout;

}

#include <iostream>

#include "Date.h"

using namespace std;

int main()

{

Date date(8, 2, 2013);

cout << date.GetDayOfTheWeek() << endl;//Friday

cout << date << endl;//8.2.2013

date += 43;

cout << date.GetDayOfTheWeek() << endl;//Saturday

cout << date << endl;//23.3.2013

date -= 23;

cout << date.GetDayOfTheWeek() << endl;//Thursday

cout << date << endl;//28.2.2013

return 0;

}

Знание кода Жиденко Н.Н. Отметки

Контрольные вопросы Жиденко Н.Н. Отметки

Автор Преподаватель

Жиденко Н.Н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (подпись) Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись \_\_\_\_\_\_\_\_ Дата