МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

**Институт математики, естественных и компьютерных наук**

**Информатика и вычислительная техника**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Разработка классов на С++.

Дисциплина: «Объектно-ориентированное программирование»

Направление подготовки: 09.03.01. Информатика и вычислительная техника

|  |  |
| --- | --- |
| Руководитель | Сорокин А.Н. |
| Выполнили студенты | Пчелкина О.С. |
| Группа, курс | ВМ-41 |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись преподавателя)* |

Вологда

2022 г.

**Цель работы**: научиться разрабатывать классы на языке С++.

# Вариант 1

1. Разработать класс CDate для работы с датами. Класс должен содержать следующие элементы.

Поля (скрытые):

день

месяц

год

Методы:

конструктор по умолчанию, присваивающий полям значение текущей (системной) даты; конструктор, параметрами которого являются заданные день, месяц и год; конструктор-копировщик

методы-функции для задания значений даты и получения дня, месяца и года даты

метод-функция для вывода даты на экран в общепринятом формате «дд.мм.гггг»

методы-функции для прибавления и вычитания из даты целого числа дней

2. Реализовать для разработанного класса методы-операторы:

- (минус) - для вычитания из одной даты другой. При этом результат должен измеряться в днях

++ и -- (два знака плюс или минус подряд) - для увеличения или уменьшения даты на 1 день

== (два знака равенства подряд) - для сравнения двух дат, результатом этой операции должен быть 0, если даты неравны, и любое ненулевое значение в противном случае

3. Разработать программу, которая демонстрировала бы работоспособность всех методов данного класса.

Код программы:

#include <iostream>

#include <ctime>

#include <chrono>

using namespace std;

class CDate

{

private:

int \_day;

int \_month;

int \_year;

time\_t \_date;

public:

CDate()

{

struct tm\* tim;

\_date = time(NULL);

tim = localtime(&\_date);

\_day = tim->tm\_mday;

\_month = tim->tm\_mon + 1;

\_year = tim->tm\_year + 1900;

}

CDate(int day, int month, int year)

{

\_day = day;

\_month = month;

\_year = year;

tm time = tm();

time.tm\_mday = \_day;

time.tm\_mon = \_month - 1;

time.tm\_year = \_year - 1900;

\_date = mktime(&time);

}

CDate(CDate& c)

{

\_day = c.\_day;

\_month = c.\_month;

\_year = c.\_year;

tm time = tm();

time.tm\_mday = \_day;

time.tm\_mon = \_month - 1;

time.tm\_year = \_year - 1900;

\_date = mktime(&time);

}

int operator - (CDate &decDate)

{

time\_t newDate = \_date;

newDate -= decDate.\_date;

return newDate/84700;

}

CDate operator ++ (int)

{

\_date += 86400;

struct tm\* tim = localtime(&\_date);

int day = tim->tm\_mday;

int month = tim->tm\_mon + 1;

int year = tim->tm\_year + 1900;

CDate result(day, month, year);

return result;

}

CDate operator -- (int)

{

\_date -= 86400;

struct tm\* tim = localtime(&\_date);

int day = tim->tm\_mday;

int month = tim->tm\_mon + 1;

int year = tim->tm\_year + 1900;

CDate result(day, month, year);

return result;

}

bool operator == (CDate &date)

{

return date.\_date == \_date;

}

int GetDay() const

{

return \_day;

}

int GetMonth() const

{

return \_month;

}

int GetYear() const

{

return \_year;

}

void SetDay(int day)

{

\_day = day;

tm time = tm();

time.tm\_mday = \_day;

time.tm\_mon = \_month - 1;

time.tm\_year = \_year - 1900;

\_date = mktime(&time);

}

void SetMonth(int month)

{

\_month = month;

tm time = tm();

time.tm\_mday = \_day;

time.tm\_mon = \_month - 1;

time.tm\_year = \_year - 1900;

\_date = mktime(&time);

}

void SetYear(int year)

{

\_year = year;

tm time = tm();

time.tm\_mday = \_day;

time.tm\_mon = \_month - 1;

time.tm\_year = \_year - 1900;

\_date = mktime(&time);

}

void AddDays(int countDay)

{

\_date += countDay \* 86400;

struct tm\* tim = localtime(&\_date);

\_day = tim->tm\_mday;

\_month = tim->tm\_mon + 1;

\_year = tim->tm\_year + 1900;

}

void DecDays(int countDay)

{

\_date -= countDay \* 86400;

struct tm\* tim = localtime(&\_date);

\_day = tim->tm\_mday;

\_month = tim->tm\_mon + 1;

\_year = tim->tm\_year + 1900;

}

void Print()

{

struct tm\* u;

char s2[100] = { 0 };

u = localtime(&\_date);

strftime(s2, 40, "%d.%m.%Y", u);

printf("%s", s2);

}

};

Программа, демонстрирующая работу класса:

#include <iostream>

#include "CDate.cpp"

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

cout << "присваивание системной даты: ";

CDate sysDate;

sysDate.Print();

cout << endl;

cout << "задаем дату через конструктор: ";

CDate setDate(1, 3, 2023);

setDate.Print();

cout << endl;

cout << "задайте день: ";

int day = 0;

cin >> day;

sysDate.SetDay(day);

cout << "новая дата: ";

sysDate.Print();

cout << endl;

cout << "задайте месяц: ";

int month = 0;

cin >> month;

sysDate.SetMonth(month);

cout << "новая дата: ";

sysDate.Print();

cout << endl;

cout << "задайте год: ";

int year = 0;

cin >> year;

sysDate.SetYear(year);

cout << "новая дата: ";

sysDate.Print();

cout << endl;

cout << "вывод дня: " << sysDate.GetDay() << endl;

cout << "вывод месяца: " << sysDate.GetMonth() << endl;

cout << "вывод года: " << sysDate.GetYear() << endl;

cout << "текущая дата: ";

setDate.Print();

cout << endl;

cout << "введите количество дней для прибавления: ";

int addDay = 0;

cin >> addDay;

cout << "новая дата: ";

setDate.AddDays(addDay);

setDate.Print();

cout << endl;

cout << "текущая дата: ";

setDate.Print();

cout << endl;

cout << "введите количество дней для уменьшения : ";

int decDay = 0;

cin >> decDay;

cout << "новая дата: ";

setDate.DecDays(decDay);

setDate.Print();

cout << endl;

cout << "вычитание дат: ";

CDate decDate1(1, 3, 2022);

decDate1.Print();

cout << " - ";

CDate decDate2(26, 2, 2022);

decDate2.Print();

cout << " = " << abs(decDate1 - decDate2);

cout << endl;

cout << "текущая дата: ";

setDate.Print();

cout << endl;

cout << "инкрементиование даты: ";

setDate++;

setDate.Print();

cout << endl;

cout << "текущая дата: ";

sysDate.Print();

cout << endl;

cout << "декрементирование даты: ";

sysDate--;

sysDate.Print();

cout << endl;

cout << "сравнение дат: ";

setDate.Print();

cout << " == ";

sysDate.Print();

bool res = sysDate == setDate;

cout << " = " << res;

cout << endl;

cout << "сравнение дат: ";

CDate newDate1(1, 3, 2000);

newDate1.Print();

cout << " == ";

CDate newDate2(1, 3, 2000);

newDate2.Print();

bool result = newDate1 == newDate2;

cout << " = " << result;

cout << endl;

}

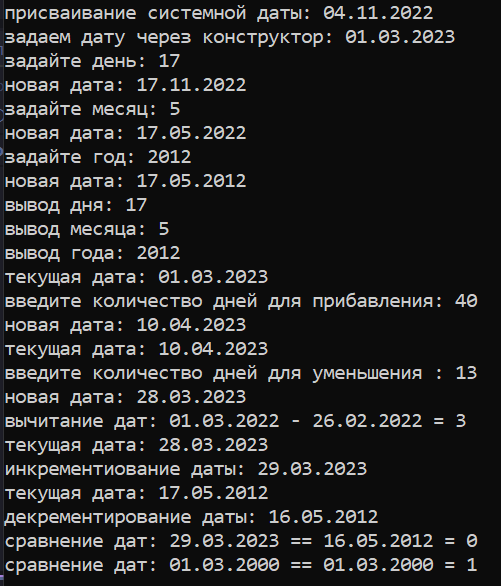


Рис. 1 – Работа программы

# Контрольные вопросы

1. Что понимается под термином «класс»? Каково соотношение понятий «класс» и «экземпляр класса»?

*Класс — это тип данных, определяемый пользователем, который похож на структуру или запись, так как содержит поля с данными, но кроме полей в описание класса входят также функции, которые предназначены для обработки полей.*

*Переменные класса иначе называются объектами или экземплярами.*

2. Что понимается под термином «конструктор»? В каких случаях вызывается конструктор копирования и конструктор преобразования класса?

*Конструктор — это особый метод, который выполняется автоматически при создании объекта. При этом конструктор имеет то же самое имя, что и класс.*

*Конструктор-копировщик чаще всего вызывают при передаче в функцию параметра-значения, который является объектом.*

*Конструктор-преобразования вызыватеся автоматически для преобразования значений других типов в объекты данного класса*

*В C++ полная копия объекта нужна в трех случаях:*

*• В момент объявления нового объекта и его инициализации данными другого объекта с помощью оператора =.*

*• Когда нужно передать объект в функцию как параметр-значение. В этом случае создается полная копия объекта.*

*• Когда нужно вернуть объект из функции по значению. В этом случае также создается полная копия объекта.*

3. Что понимается под термином «деструктор»? В каких случаях применяется деструктор класса?

*Деструктор — это особый метод, который выполняется автоматически при уничтожении объекта.*

*Деструкторы имеют то же самое имя, что и класс, но со знаком ~*

*Деструктор вызывается в том случае когда программа покидает область видимости, после этого происходит чистка полей класса.*

**Вывод:** в ходе лабораторной работы ознакомились с основными принципами ООП, научились создавать классы, изучили работу конструкторов, деструкторов, перегрузки операторов. Разработали класс на языке С++ и программу для демонстрации работы класса.

ДЕКРЕМЕНТ ИНКРЕМЕНТ, ПРЕФИКСНЫЕ, ВОПРОСЫ 2,3