**ФГБОУ ВО**

**Национальный исследовательский университет**

**«МЭИ»**

**Лабораторная работа № 3**

по курсу

«Технология программирования»

**ПРОСТЕЙШИЕ КЛАССЫ, ОДИНОЧНОЕ НАСЛЕДОВАНИЕ**

Вариант 8

**Выполнил:**

студент группы Аэ-21-23

Нестеров А.С.

**Москва, 2024**

1. **Постановка задачи**

*Задача:* Создать класс. Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы, свойства. Методы и свойства должны обеспечивать непротиворечивый, полный, минимальный и удобный интерфейс класса. При возникновении ошибок должны выбрасываться исключения. В программе должна выполняться проверка всех разработанных элементов класса. Создать дочерний класс. Реализовать задание с помощью указателей: (в классе поля должны быть объявлены динамическими, в основной программе объявить переменную - указатель на класс).

*Условие*: Составить описание класса для представления даты. Предусмотреть возможности установки даты и изменения ее отдельных полей (год, месяц, день) с проверкой допустимости вводимых значений. В случае недопустимых значений выбрасываются исключения. Создать методы изменения даты на заданное количество дней, месяцев и лет. Написать программу, демонстрирующую все разработанные элементы класса. Создать дочерний класс, определяющий по дате день недели.

* **Класс Date:**
  + Методы для создания объектов, валидации даты, установки и получения значений года, месяца и дня, а также добавления дней, месяцев и лет.
  + Исключение выбрасывается при попытке установить недопустимую дату.
* **Класс Weekday (дочерний класс):**
  + Методы для получения дня недели для заданной даты с использованием алгоритма Зеллера.

**Математические формулы:**

* Алгоритм Зеллера для вычисления дня недели:

h=(d+⌊13(m+1)5⌋+k+⌊k4⌋+⌊j4⌋−2j)mod  7*h*=(*d*+⌊513(*m*+1)​⌋+*k*+⌊4*k*​⌋+⌊4*j*​⌋−2*j*)mod7

где:

* + y*y* - год
  + m*m* - месяц (январь и февраль рассматриваются как 13 и 14 предыдущего года)
  + d*d* - день
  + k*k* - последние две цифры года
  + j*j* - первые две цифры года
  + h*h* - день недели (0 - суббота, 1 - воскресенье, ..., 6 - пятница)

**Функции программы:**

* Создать дату
* Получить день недели для заданной даты
* Завершить программу

**Входные данные:** Год, месяц и день.

**Выходные данные:** День недели для заданной даты.

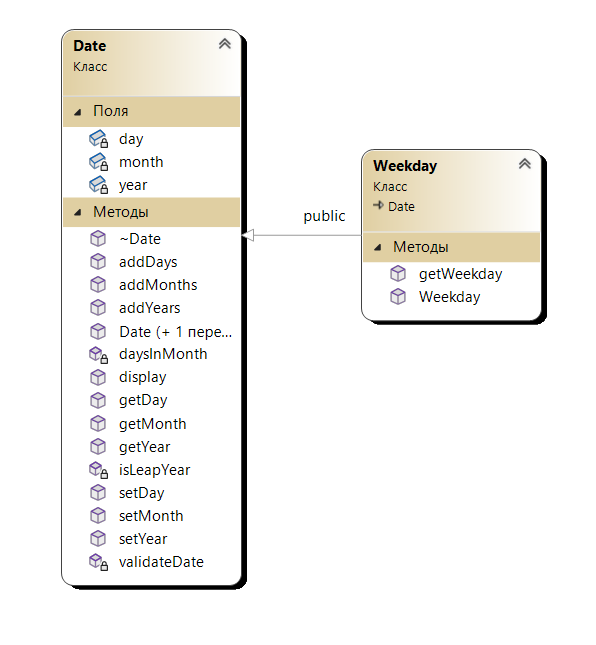
**Ограничения:** Год должен быть положительным (и меньше 10000). Месяц должен быть в диапазоне от 1 до 12. День должен быть допустимым для указанного месяца и года.

**2.1 Разработка структуры приложения**

В данной программе описаны два класса:

* **class Date** – класс для представления даты, включая методы для валидации и манипуляции датами.
* **class Weekday** – дочерний класс, который расширяет функциональность класса Date, добавляя метод для определения дня недели.

**2.2 Описание полей и методов классов**



*Рис. 1. Диаграмма классов*

**class Date:**

**Поля:**

* int\* year: указатель на целочисленный тип, представляющий год.
* int\* month: указатель на целочисленный тип, представляющий месяц.
* int\* day: указатель на целочисленный тип, представляющий день.

**Методы:**

* Date(): конструктор без параметров, инициализирующий дату на 20 декабря 2005 года.
* Date(int year, int month, int day): конструктор с параметрами, устанавливающий дату и выполняющий валидацию.
* void setYear(int year): метод для установки года с валидацией.
* void setMonth(int month): метод для установки месяца с валидацией.
* void setDay(int day): метод для установки дня с валидацией.
* int getYear(): метод для получения года.
* int getMonth(): метод для получения месяца.
* int getDay(): метод для получения дня.
* void addDays(int days): метод для добавления дней к текущей дате.
* void addMonths(int months): метод для добавления месяцев к текущей дате, с учетом перехода на следующий год.
* void addYears(int years): метод для добавления лет к текущей дате, с учетом перехода на следующий месяц.
* void display(): метод для отображения даты в формате "дд.мм.гггг".
* ~Date(): деструктор, освобождающий память.

**2.3 Разработка пользовательского интерфейса**

**Категории пользователей:** Студенты и преподаватели, желающие протестировать функциональность программы.

**Функции пользователя (интерфейс):**

* Ввод года, месяца и дня.
* Установка значений года, месяца и дня.
* Добавление дней, месяцев и лет к текущей дате.
* Отображение текущей даты.
* Получение дня недели для заданной даты.
* Завершение программы.

Интерфейс программы реализован в виде текстового меню, которое позволяет пользователю выбирать операции. Пример интерфейса:

----------------------------------------------------------------

Выберите операцию:

1. Создать дату (по умолчанию 20.12.2005)

2. Создать дату (ввести год, месяц, день)

3. Установить год

4. Установить месяц

5. Установить день

6. Добавить дни

7. Добавить месяцы

8. Добавить годы

9. Отобразить дату

10. Получить день недели

0. Выход

Ваш выбор?

*Рис.2. Диалог с пользователем*

При выборе каждой операции программа запрашивает необходимые данные и выполняет соответствующие действия. Если вводимые данные некорректны (например, нецелые числа или недопустимые значения), программа сообщает об ошибке и предлагает повторить ввод.

**3. Реализация и тестирование приложения**

**Описание разработанной программы**

Данная программа написана на языке C++ и включает два класса: class Date для управления датами и class Weekday для вычисления дня недели. Основная задача программы — демонстрация работы с классами, обработка исключений и валидация данных.

**Тестирование программы**

Тестирование программы включает:

* **Нормальные условия:** ввод корректных данных и успешное вычисление дня недели.
* **Исключительные условия:** ввод некорректных данных, таких как недопустимые даты или неверные форматы ввода.

**Таблица тестирования**

Вот таблица тестирования для функций класса Date, включающая различные сценарии, чтобы обеспечить всестороннюю проверку функциональности программы:

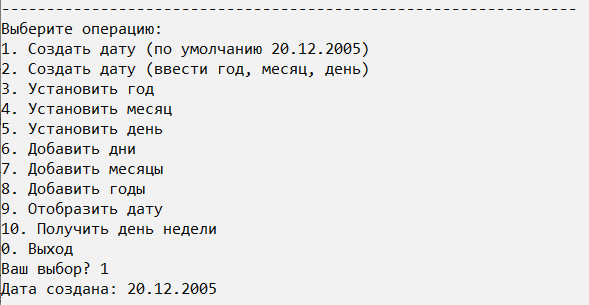
Табл. 1. Тесты программы

| **№ теста** | **Тестируемая функция** | **Входные данные** | **Ожидаемый результат** | **Цель теста** |
| --- | --- | --- | --- | --- |
| 1 | Date() | - | Дата: 20.12.2005 | Проверка конструктора без параметров |
| 2 | Date(int year, int month, int day) | 2023, 2, 28 | Дата: 28.02.2023 | Проверка корректной инициализации даты |
| 3 | Date(int year, int month, int day) | 2023, 2, 29 | Исключение: "Недопустимый день для данного месяца." | Проверка валидации даты для несуществующего дня |
| 4 | setYear(int year) | 2024 | Успешно устанавливает год на 2024 | Проверка установки года с валидацией |
| 5 | setYear(int year) | 0 | Исключение: "Год должен быть положительным." | Проверка валидации года при установке |
| 6 | setMonth(int month) | 5 | Успешно устанавливает месяц на 5 | Проверка установки месяца с валидацией |
| 7 | setMonth(int month) | 13 | Исключение: "Месяц должен быть от 1 до 12." | Проверка валидации месяца при установке |
| 8 | setDay(int day) | 15 | Успешно устанавливает день на 15 | Проверка установки дня с валидацией |
| 9 | setDay(int day) | 32 | Исключение: "Недопустимый день для данного месяца." | Проверка валидации дня при установке |
| 10 | getYear() | - | 2005 | Проверка получения года |
| 11 | getMonth() | - | 12 | Проверка получения месяца |
| 12 | getDay() | - | 20 | Проверка получения дня |
| 13 | addDays(int days) | 10 | Дата: 30.12.2005 | Проверка добавления дней |
| 14 | addDays(int days) | -5 | Дата: 15.12.2005 | Проверка вычитания дней |
| 15 | addMonths(int months) | 3 | Дата: 20.03.2006 | Проверка добавления месяцев |
| 16 | addMonths(int months) | -1 | Дата: 20.11.2005 | Проверка вычитания месяцев |
| 17 | addYears(int years) | 5 | Дата: 20.12.2010 | Проверка добавления лет |
| 18 | addYears(int years) | -6 | Дата: 20.12.2004 | Проверка вычитания лет |
| 19 | display() | - | "20.12.2005" | Проверка корректного отображения даты |
| 20 | ~Date() | - | Освобождение памяти | Проверка работы деструктора |

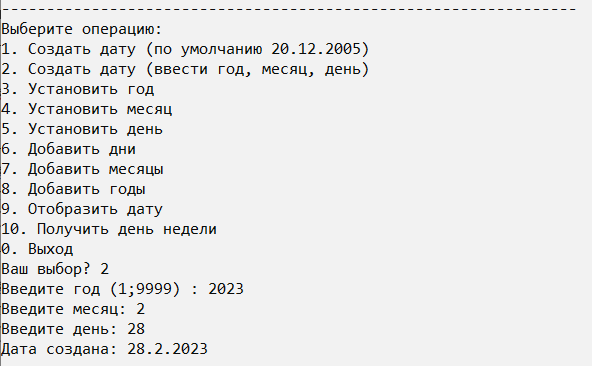
**Объяснение:**

* **Тесты 1-3** проверяют корректность конструктора и его валидацию.
* **Тесты 4-9** проверяют методы установки года, месяца и дня, включая валидацию.
* **Тесты 10-12** проверяют методы получения значений года, месяца и дня.
* **Тесты 13-18** проверяют методы добавления дней, месяцев и лет.
* **Тест 19** проверяет корректность отображения даты.
* **Тест 20** проверяет освобождение памяти при уничтожении объекта.

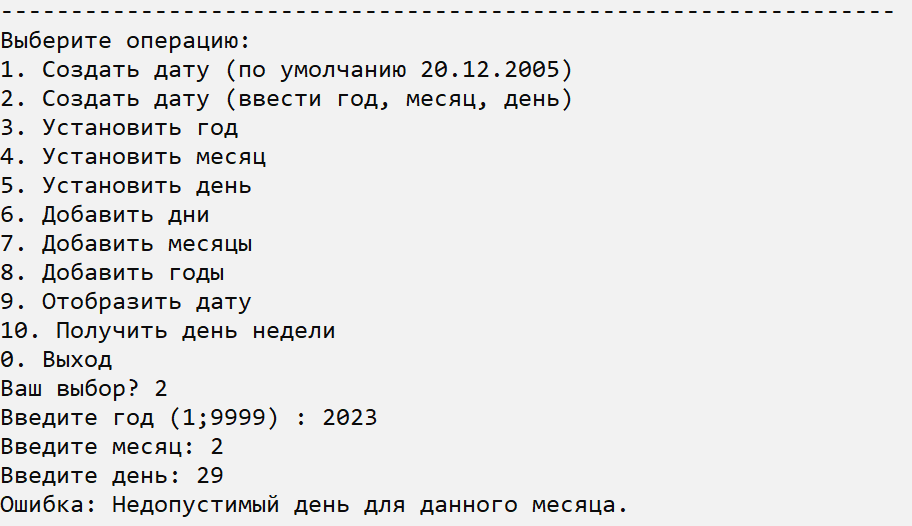
Эта таблица тестирования охватывает основные функции класса Date и помогает убедиться в их корректной работе и обработке исключений.



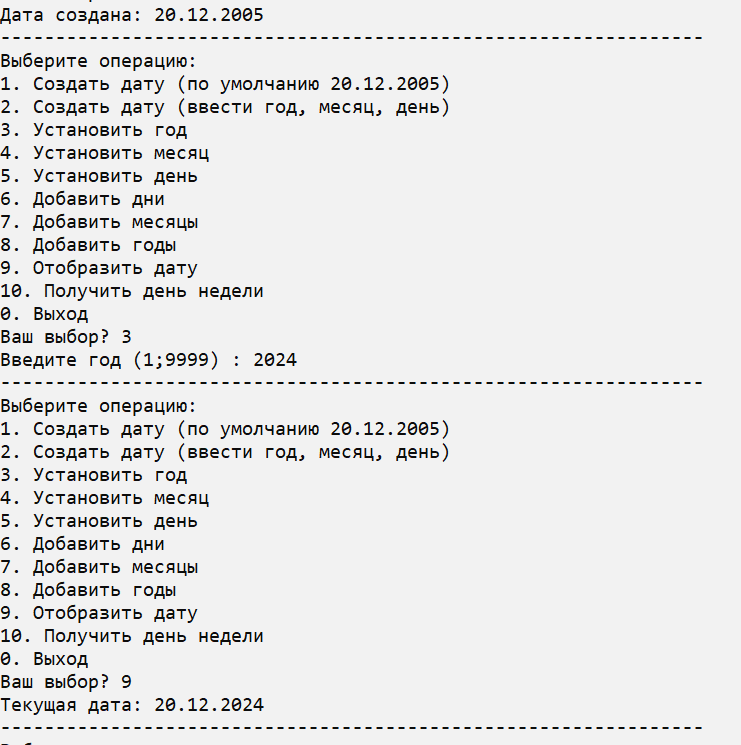
*Рис.3. Тест 1*



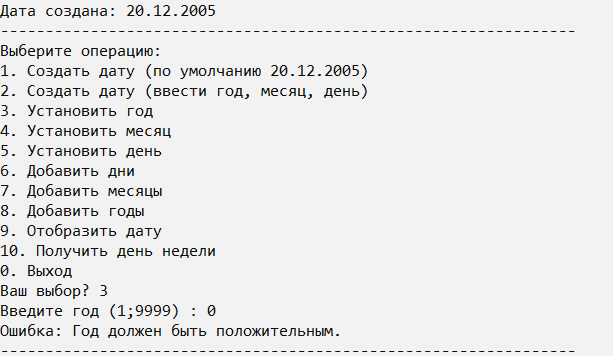
*Рис.4 Тест 2*



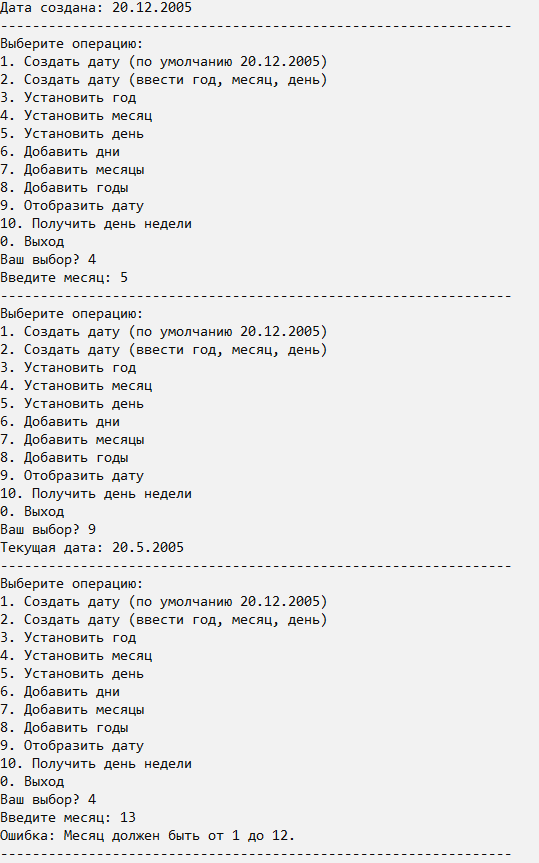
*Рис.5 Тест 3*



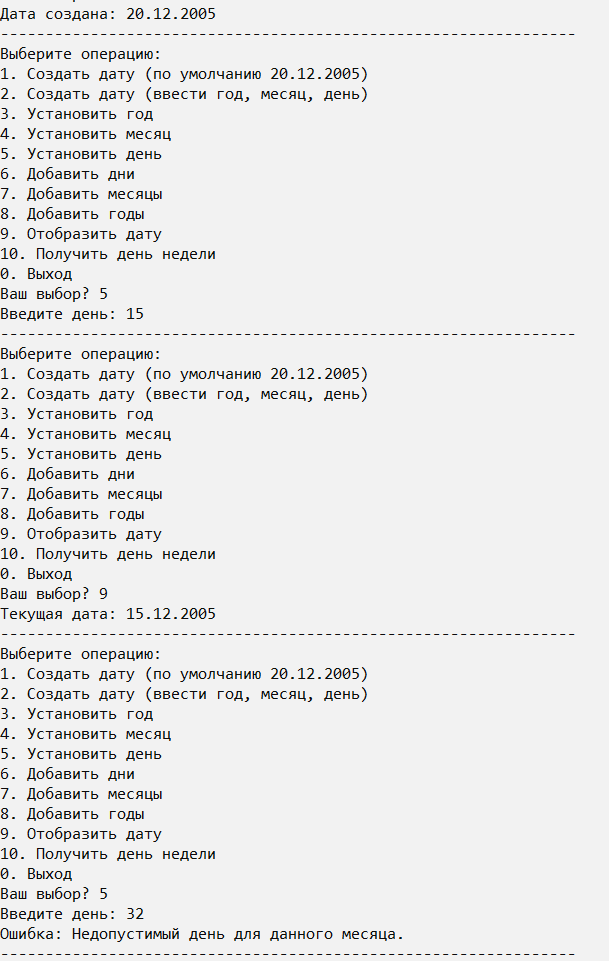
*Рис.6 Тест 4*



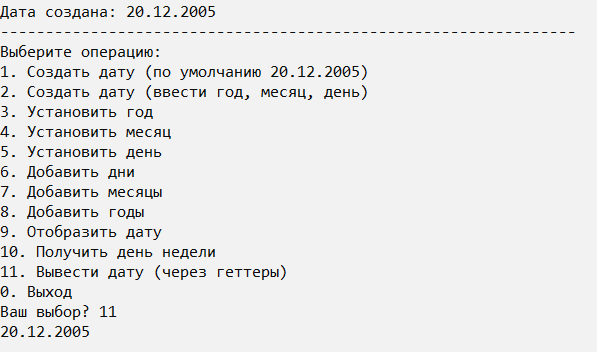
*Рис.7 Тест 5*



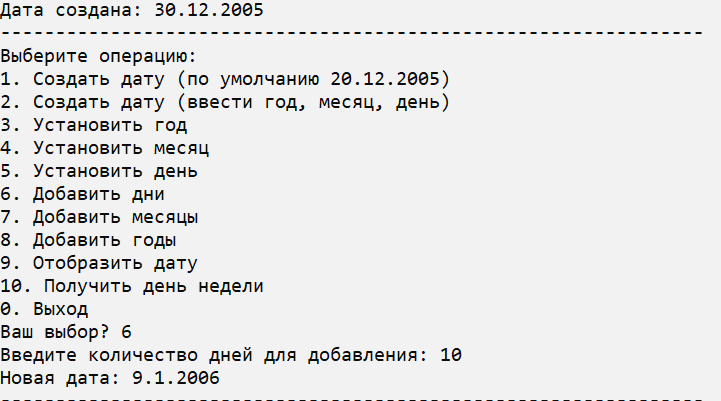
*Рис.8 Тест 6,7*



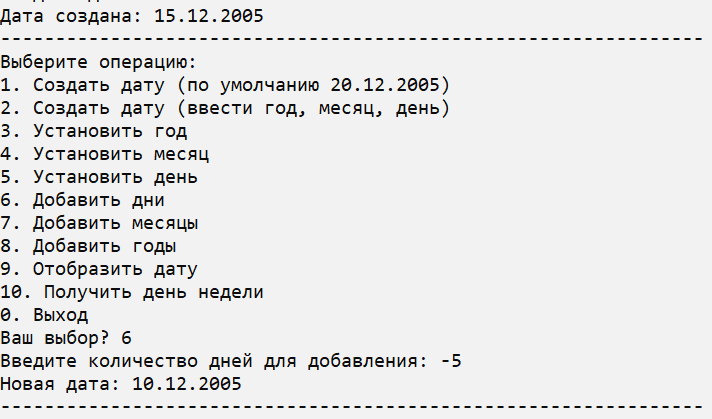
*Рис.9 Тест 8,9*



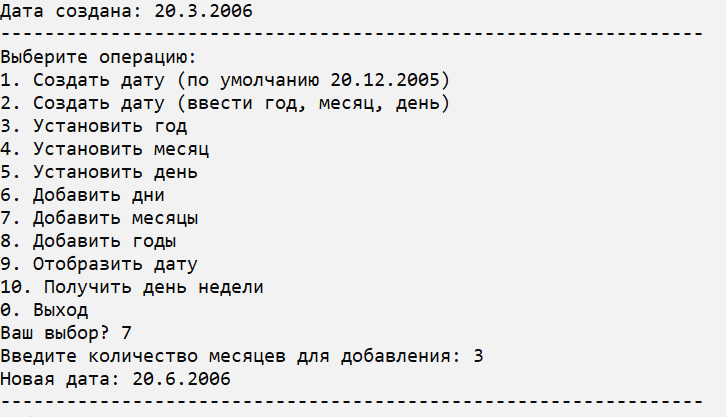
*Рис.10 Тест 10, 11, 12*



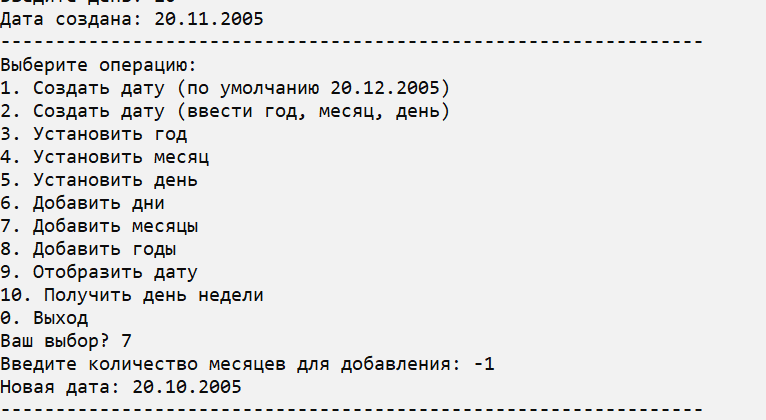
*Рис.11 Тест 13*



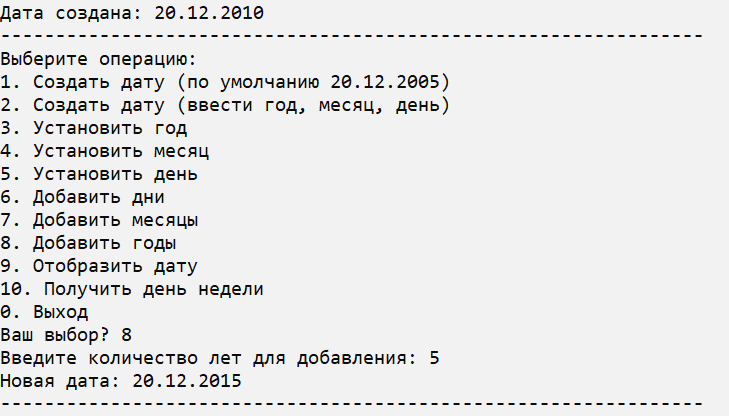
*Рис.12 Тест 14*



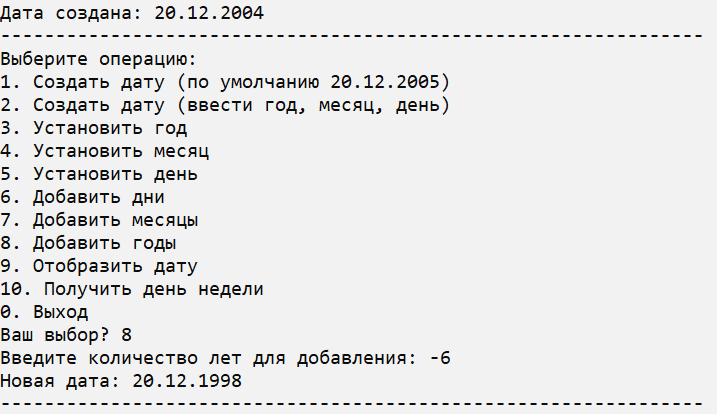
*Рис.13 Тест 15*



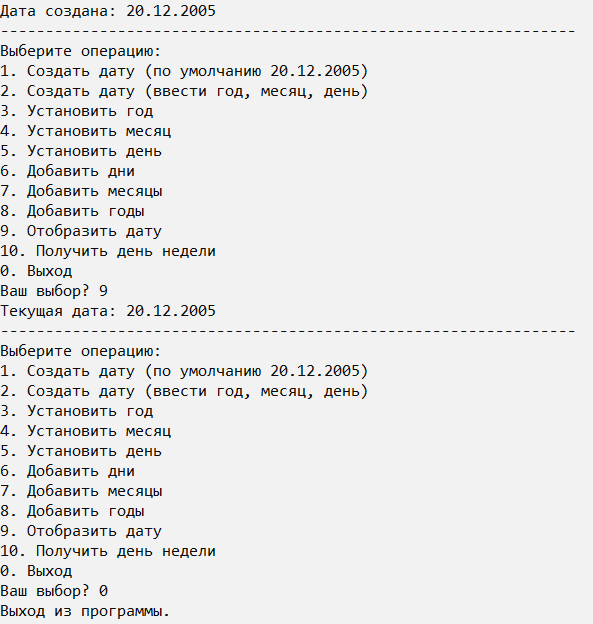
*Рис.14 Тест 16*



*Рис.15 Тест 17*



*Рис.16 Тест 18*



*Рис.17 Тест 19-20*

Таблица тестирования для методов класса Weekday, которая включает различные сценарии для обеспечения всесторонней проверки функциональности программы:

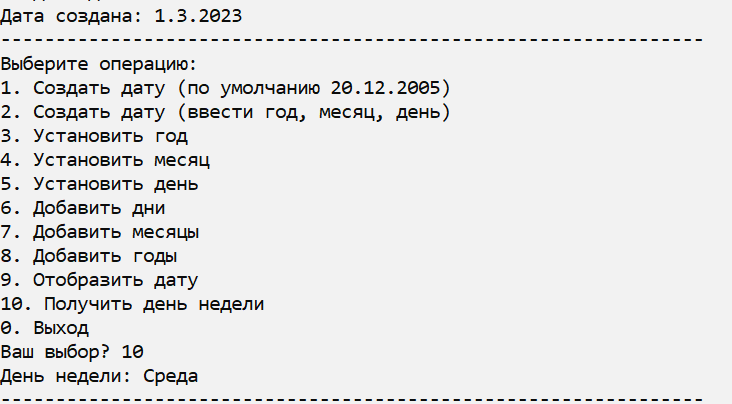
Таблица 1 – Тесты программы

| **№ теста** | **Тестируемая функция** | **Входные данные** | **Ожидаемый результат** | **Цель теста** |
| --- | --- | --- | --- | --- |
| 21 | Weekday(int year, int month, int day) | 2023, 3, 1 | Объект Weekday создан | Проверка корректной инициализации объекта |
| 22 | getWeekday() | 2023, 3, 1 | "Среда" | Проверка правильного вычисления дня недели |
| 23 | getWeekday() | 2023, 2, 28 | "Вторник" | Проверка правильного вычисления дня недели для конца месяца |
| 24 | getWeekday() | 2024, 2, 29 | "Четверг" | Проверка правильного вычисления дня недели для високосного года |
| 25 | getWeekday() | 2000, 1, 1 | "Суббота" | Проверка правильного вычисления дня недели для начала века |
| 26 | getWeekday() | 1900, 1, 1 | "Понедельник" | Проверка правильного вычисления дня недели для года, не являющегося високосным |
| 27 | getWeekday() | 2023, 12, 31 | "Воскресенье" | Проверка правильного вычисления дня недели для конца года |
| 28 | getWeekday() | 2023, 11, 5 | "Воскресенье" | Проверка правильного вычисления дня недели для даты, совпадающей с переходом на зимнее время |
| 29 | getWeekday() | 2023, 4, 1 | "Суббота" | Проверка правильного вычисления дня недели для даты в начале месяца |
| 30 | getWeekday() | 2024, 3, 1 | "Пятница" | Проверка правильного вычисления дня недели для даты в начале месяца високосного года |

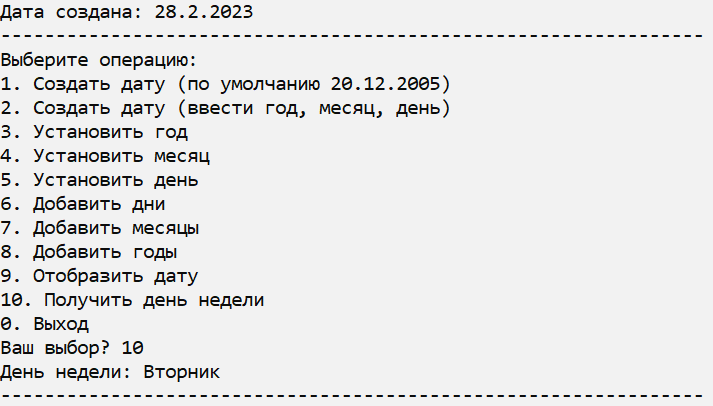
**Объяснение:**

* **Тесты 21** проверяют корректность создания объекта Weekday с заданной датой.
* **Тесты 22-30** проверяют правильность вычисления дня недели для различных дат, включая високосные годы, начало и конец месяца и года, а также важные исторические даты.

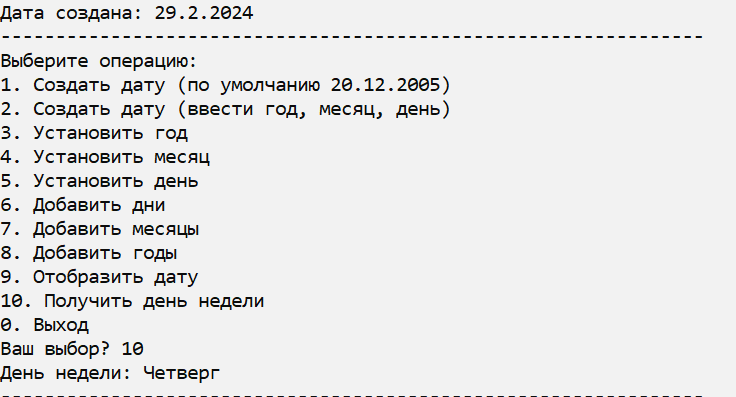
Эта таблица тестирования охватывает основные функции класса Weekday и помогает убедиться в их корректной работе.



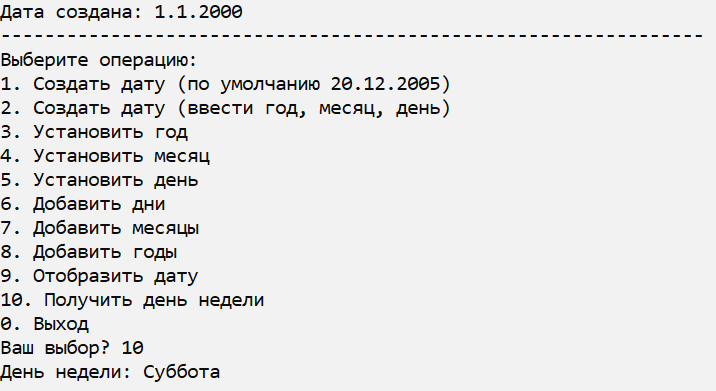
*Рис.18 Тест 21,22*



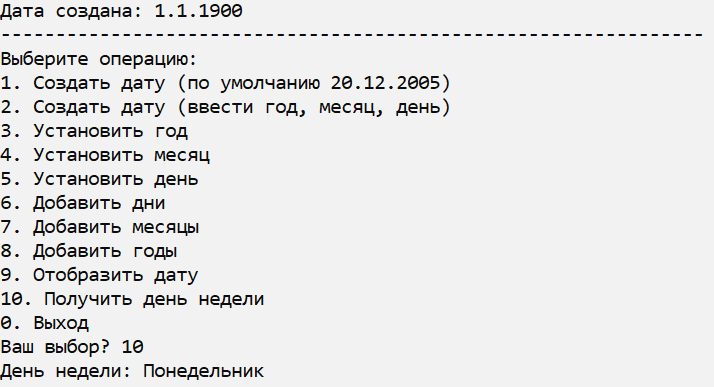
*Рис.19 Тест 23*



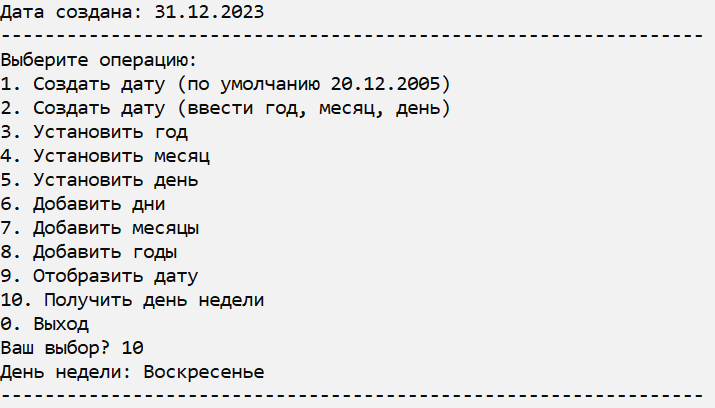
*Рис.20 Тест 24*



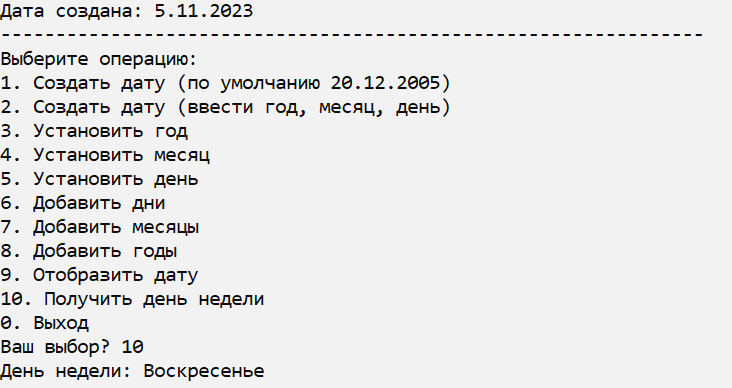
*Рис.21 Тест 25*



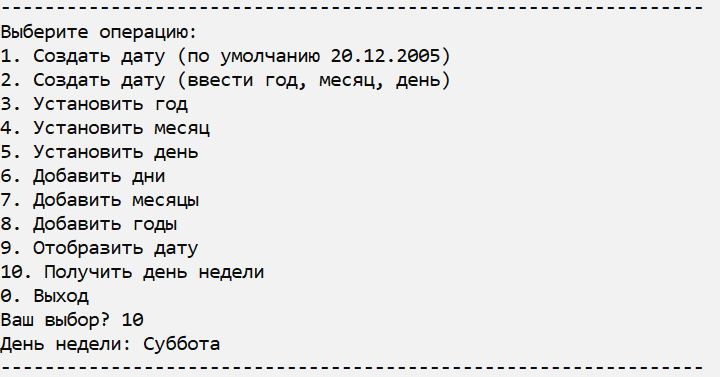
*Рис.22 Тест 26*



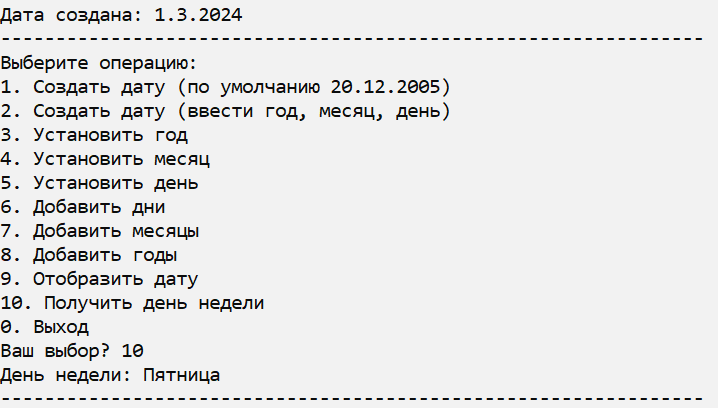
*Рис.23 Тест 27*



*Рис.24 Тест 28*



*Рис.25 Тест 29*



*Рис.26 Тест 30*

**Вывод**

В данной лабораторной работе изучены принципы работы с классами в C++, включая создание, наследование и обработку исключений. Классы Date и Weekday демонстрируют основные концепции объектно-ориентированного программирования и управление памятью. Программа успешно обрабатывает различные сценарии ввода и предоставляет пользователю удобный интерфейс для работы с датами.

*Приложение 1.* Код программы

#include <iostream>

#include <stdexcept>

#include <string>

#include <cmath>

#include <windows.h>

class Date {

private:

int\* year; // Указатель на целое число, представляющее год

int\* month; // Указатель на целое число, представляющее месяц

int\* day; // Указатель на целое число, представляющее день

// Метод для валидации даты

void validateDate(int year, int month, int day) {

if (year < 1) throw std::invalid\_argument("Год должен быть положительным.");

if (year > 9999) throw std::invalid\_argument("Год должен быть меньше 9999.");

if (month < 1 || month > 12) throw std::invalid\_argument("Месяц должен быть от 1 до 12.");

if (day < 1 || day > daysInMonth(year, month)) throw std::invalid\_argument("Недопустимый день для данного месяца.");

}

// Метод для получения количества дней в месяце

int daysInMonth(int year, int month) {

if (month == 2) {

return (isLeapYear(year)) ? 29 : 28; // Февраль

}

return (month == 4 || month == 6 || month == 9 || month == 11) ? 30 : 31; // Апрель, Июнь, Сентябрь, Ноябрь

}

// Метод для проверки високосного года

bool isLeapYear(int year) {

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

}

public:

// Конструктор без параметров, инициализирующий дату на 20 декабря 2005 года

Date() {

year = new int(2005);

month = new int(12);

day = new int(20);

}

// Конструктор с параметрами для установки даты

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

validateDate(year, month, day); // Валидация даты

this->year = new int(year);

this->month = new int(month);

this->day = new int(day);

}

// Метод для установки года

void setYear(int year) {

validateDate(year, \*month, \*day);

\*(this->year) = year;

}

// Метод для установки месяца

void setMonth(int month) {

validateDate(\*year, month, \*day);

\*(this->month) = month;

}

// Метод для установки дня

void setDay(int day) {

validateDate(\*year, \*month, day);

\*(this->day) = day;

}

// Метод для получения года

int getYear() { return \*year; }

// Метод для получения месяца

int getMonth() { return \*month; }

// Метод для получения дня

int getDay() { return \*day; }

// Метод для добавления дней

void addDays(int days) {

if (days >= 0) {

// Добавление дней

for (int i = 0; i < days; ++i) {

(\*day)++;

if (\*day > daysInMonth(\*year, \*month)) {

\*day = 1;

(\*month)++;

if (\*month > 12) {

\*month = 1;

(\*year)++;

}

}

}

}

else {

// Вычитание дней

for (int i = 0; i < -days; ++i) {

(\*day)--;

if (\*day < 1) {

(\*month)--;

if (\*month < 1) {

\*month = 12;

(\*year)--;

}

\*day = daysInMonth(\*year, \*month); // Устанавливаем день на последний день предыдущего месяца

}

}

}

if (\*year < 1)

{

std::cout << "Изменить дату на заданное число дней невозможно, так как год выйдет неположительным. Дата останется прежней." << std::endl;

addDays(-1 \* days);

}

}

// Метод для добавления месяцев

void addMonths(int months) {

\*month += months;

// Обработка переполнения для положительных месяцев

while (\*month > 12) {

\*month -= 12;

(\*year)++;

}

// Обработка переполнения для отрицательных месяцев

while (\*month < 1) {

\*month += 12;

(\*year)--;

}

if (\*year < 1)

{

std::cout << "Изменить дату на заданное число месяцев невозможно, так как год выйдет неположительным. Дата останется прежней." << std::endl;

addMonths(-1 \* months);

}

}

// Метод для добавления лет

void addYears(int years) {

\*year += years;

if (\*day > daysInMonth(\*year, \*month)) {

\*day = daysInMonth(\*year, \*month);

}

if (\*year < 1)

{

std::cout << "Изменить дату на заданное число лет невозможно, так как год выйдет неположительным. Дата останется прежней." << std::endl;

addYears(-1 \* years);

}

}

// Метод для отображения даты в формате "дд.мм.гггг"

void display() {

std::cout << \*day << "." << \*month << "." << \*year << std::endl;

}

// Деструктор для освобождения памяти

~Date() {

delete year;

delete month;

delete day;

}

};

class Weekday : public Date {

public:

// Конструктор, который принимает дату и вызывает конструктор родительского класса

Weekday(int year, int month, int day) : Date(year, month, day) {}

// Метод, который возвращает день недели для заданной даты

std::string getWeekday() {

// Алгоритм Зеллера для вычисления дня недели

int y = getYear();

int m = getMonth();

int d = getDay();

// Если месяц январь или февраль, то рассматриваем его как 13 или 14 предыдущего года

if (m < 3) {

m += 12;

y--;

}

int k = y % 100; // Последние две цифры года

int j = y / 100; // Первые две цифры года

// Формула Зеллера

int h = (d + (13 \* (m + 1)) / 5 + k + (k / 4) + (j / 4) - (2 \* j)) % 7;

// Преобразуем результат в день недели

// h = 0 - суббота, 1 - воскресенье, 2 - понедельник, ..., 6 - пятница

switch (h) {

case 0: return "Суббота";

case 1: return "Воскресенье";

case 2: return "Понедельник";

case 3: return "Вторник";

case 4: return "Среда";

case 5: return "Четверг";

case 6: return "Пятница";

default: return ""; // Не должно быть

}

}

};

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

system("color F0"); // консоль белого цвета

Date\* date = nullptr; // Указатель на объект даты

int choice;

do {

std::cout << "----------------------------------------------------------------" << std::endl;

std::cout << "Выберите операцию:\n";

std::cout << "1. Создать дату (по умолчанию 20.12.2005)\n";

std::cout << "2. Создать дату (ввести год, месяц, день)\n";

std::cout << "3. Установить год\n";

std::cout << "4. Установить месяц\n";

std::cout << "5. Установить день\n";

std::cout << "6. Добавить дни\n";

std::cout << "7. Добавить месяцы\n";

std::cout << "8. Добавить годы\n";

std::cout << "9. Отобразить дату\n";

std::cout << "10. Получить день недели\n";

//std::cout << "11. Вывести дату (через геттеры)\n";

std::cout << "0. Выход\n";

std::cout << "Ваш выбор? ";

std::cin >> choice;

switch (choice) {

case 1: {

date = new Date(); // Создание даты по умолчанию

std::cout << "Дата создана: ";

date->display();

break;

}

case 2: {

try {

int year, month, day;

std::cout << "Введите год (1;9999) : ";

// std::cin >> year;

if (!(std:: cin >> year) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

std::cout << "Введите месяц: ";

//std::cin >> month;

if (!(std::cin >> month) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

std::cout << "Введите день: ";

//std::cin >> day;

if (!(std::cin >> day) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

date = new Date(year, month, day); // Создание даты с пользовательскими параметрами

std::cout << "Дата создана: ";

date->display();

}

catch (const std::invalid\_argument& e) {

std::cout << "Ошибка: " << e.what() << std::endl;

}

break;

}

case 3: {

if (date) {

int year;

std::cout << "Введите год (1;9999) : ";

//std::cin >> year;

if (!(std::cin >> year) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

try {

date->setYear(year);

}

catch (const std::invalid\_argument& e) {

std::cout << "Ошибка: " << e.what() << std::endl;

}

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 4: {

if (date) {

int month;

std::cout << "Введите месяц: ";

//std::cin >> month;

if (!(std::cin >> month) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

try {

date->setMonth(month);

}

catch (const std::invalid\_argument& e) {

std::cout << "Ошибка: " << e.what() << std::endl;

}

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 5: {

if (date) {

int day;

std::cout << "Введите день: ";

//std::cin >> day;

if (!(std::cin >> day) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

try {

date->setDay(day);

}

catch (const std::invalid\_argument& e) {

std::cout << "Ошибка: " << e.what() << std::endl;

}

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 6: {

if (date) {

int daysToAdd;

std::cout << "Введите количество дней для добавления: ";

//std::cin >> daysToAdd;

if (!(std::cin >> daysToAdd) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

date->addDays(daysToAdd);

std::cout << "Новая дата: ";

date->display();

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 7: {

if (date) {

int monthsToAdd;

std::cout << "Введите количество месяцев для добавления: ";

//std::cin >> monthsToAdd;

if (!(std::cin >> monthsToAdd) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

date->addMonths(monthsToAdd);

std::cout << "Новая дата: ";

date->display();

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 8: {

if (date) {

int yearsToAdd;

std::cout << "Введите количество лет для добавления: ";

//std::cin >> yearsToAdd;

if (!(std::cin >> yearsToAdd) || std::cin.get() != '\n') { // если ошибка ввода - ввели не целое число или в потоке ввода остались символы (кроме \n)

//std::cout << "Надо ввести целое число!\n" << std::endl; // вывести сообщение

std::cout << "Некорретный ввод!\n" << std::endl; // вывести сообщение

std::cin.clear(); // сбросить ошибку ввода, -> cin.fail() = 0 (cin.fail() = 1, т.к. в числовую переменную записываются символы)

while (std::cin.get() != '\n'); // очистить поток - считать все символы, пока не считается последний (т.е. \n)

break;

}

date->addYears(yearsToAdd);

std::cout << "Новая дата: ";

date->display();

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 9: {

if (date) {

std::cout << "Текущая дата: ";

date->display();

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 10: {

if (date) {

Weekday weekdayDate(date->getYear(), date->getMonth(), date->getDay());

std::cout << "День недели: " << weekdayDate.getWeekday() << std::endl;

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 11: {

if (date) {

std::cout << date->getDay() <<"." << date->getMonth() << "." << date->getYear() << std::endl;

}

else {

std::cout << "Сначала создайте дату!" << std::endl;

}

break;

}

case 0:

std::cout << "Выход из программы." << std::endl;

break;

default:

std::cout << "Неверный выбор. Попробуйте снова." << std::endl;

}

} while (choice != 0);

delete date; // Освобождение памяти

return 0;

}