**Выполнили студент 215 группы**

**Марков К.А.**

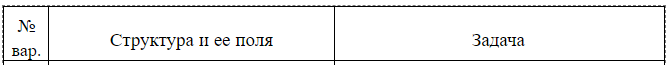
**Практическое занятие 28**

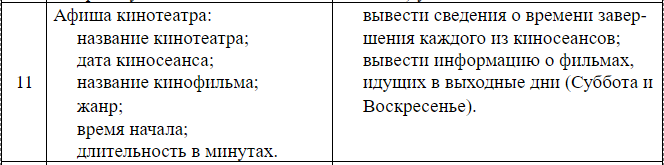
**Тема: «Тип данных – дата и время»**

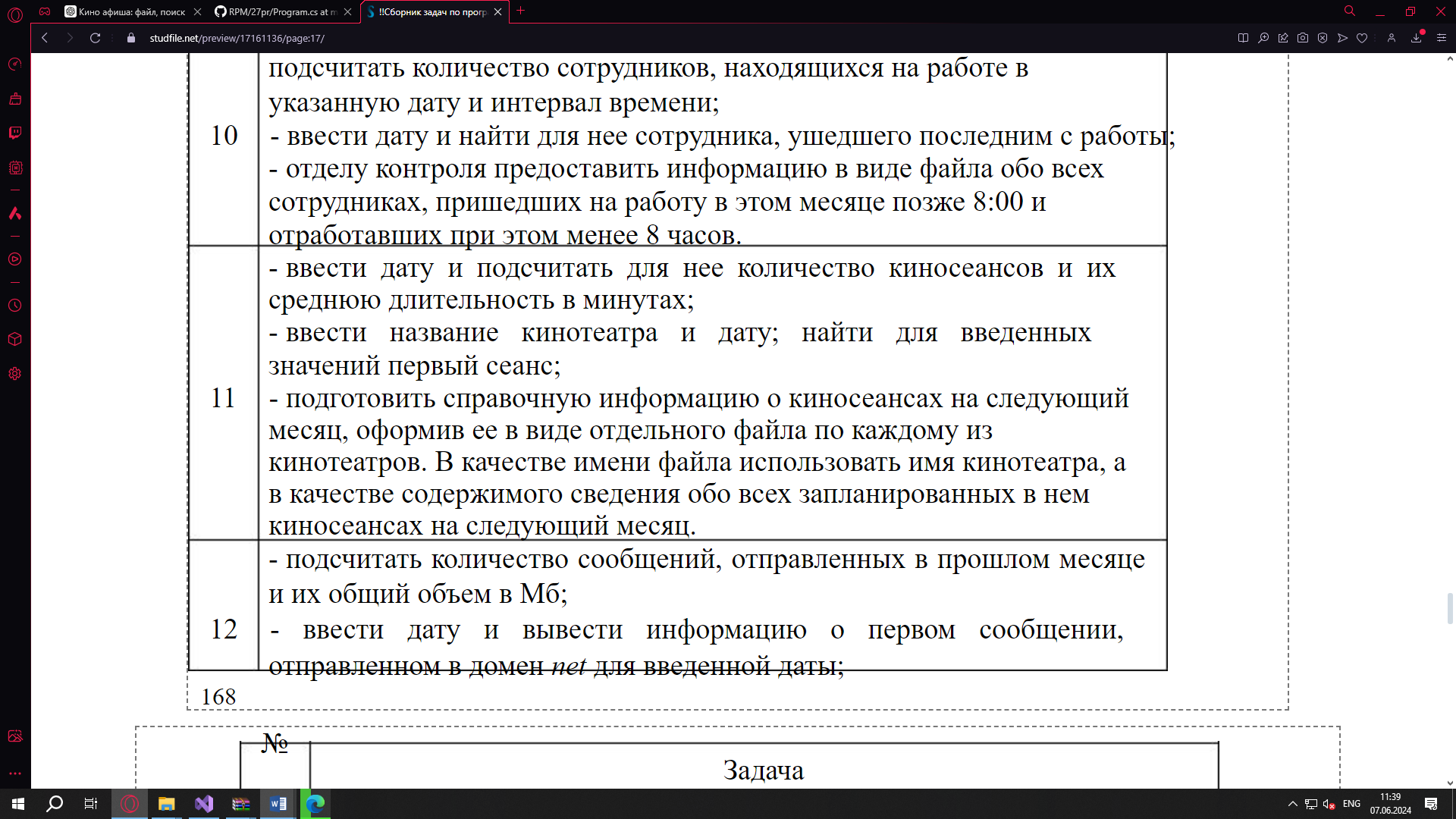
**Цель работы:** изучить набор функций, предназначенных для работы с данными типа дата и время.

**Задание:**

**Вариант 11**







// Структура CinemaPoster

struct CinemaPoster

{

public string CinemaName { get; set; }

public DateTime SessionDate { get; set; }

public string MovieTitle { get; set; }

public string Genre { get; set; }

public TimeSpan StartTime { get; set; }

public int DurationMinutes { get; set; }

// Конструктор для инициализации полей структуры

public CinemaPoster(string cinemaName, DateTime sessionDate, string movieTitle, string genre, TimeSpan startTime, int durationMinutes)

{

CinemaName = cinemaName;

SessionDate = sessionDate;

MovieTitle = movieTitle;

Genre = genre;

StartTime = startTime;

DurationMinutes = durationMinutes;

}

// Метод для отображения информации о сеансе

public void DisplayInfo()

{

Console.WriteLine($"Кинотеатр: {CinemaName}");

Console.WriteLine($"Дата сеанса: {SessionDate.ToShortDateString()}");

Console.WriteLine($"Название фильма: {MovieTitle}");

Console.WriteLine($"Жанр: {Genre}");

Console.WriteLine($"Время начала: {StartTime}");

Console.WriteLine($"Длительность: {DurationMinutes} минут");

Console.WriteLine($"Время завершения: {GetEndTime()}");

Console.WriteLine();

}

// Метод для вычисления времени завершения сеанса

public TimeSpan GetEndTime()

{

return StartTime.Add(TimeSpan.FromMinutes(DurationMinutes));

}

// Метод для преобразования структуры в строку

public override string ToString()

{

return $"{CinemaName}|{SessionDate:yyyy-MM-dd}|{MovieTitle}|{Genre}|{StartTime}|{DurationMinutes}";

}

// Метод для создания структуры из строки

public static CinemaPoster FromString(string str)

{

var parts = str.Split('|');

return new CinemaPoster(

parts[0],

DateTime.ParseExact(parts[1], "yyyy-MM-dd", CultureInfo.InvariantCulture),

parts[2],

parts[3],

TimeSpan.Parse(parts[4]),

int.Parse(parts[5])

);

}

}

class Program

{

static void Main(string[] args)

{

string filePath = "C:\\Users\\Kab-31-13\\Desktop\\28pr\\cinema\_posters.txt";

CinemaPoster[] posters = new CinemaPoster[]

{

new CinemaPoster("Кинотеатр 1", new DateTime(2024, 6, 1), "Фильм 1", "Драма", new TimeSpan(14, 30, 0), 120),

new CinemaPoster("Кинотеатр 2", new DateTime(2024, 6, 2), "Фильм 2", "Комедия", new TimeSpan(16, 0, 0), 90),

new CinemaPoster("Кинотеатр 3", new DateTime(2024, 6, 3), "Фильм 3", "Боевик", new TimeSpan(18, 45, 0), 110),

new CinemaPoster("Кинотеатр 4", new DateTime(2024, 6, 7), "Фильм 4", "Ужасы", new TimeSpan(20, 0, 0), 105),

new CinemaPoster("Кинотеатр 5", new DateTime(2024, 6, 8), "Фильм 5", "Фантастика", new TimeSpan(19, 30, 0), 95),

};

// Запись массива киноафиш в файл

WriteToFile(posters, filePath);

// Чтение массива киноафиш из файла

CinemaPoster[] loadedPosters = ReadFromFile(filePath);

// Отображение информации о времени завершения каждого сеанса

Console.WriteLine("Сведения о времени завершения каждого сеанса:\n");

foreach (var poster in loadedPosters)

{

poster.DisplayInfo();

}

// Отображение информации о фильмах, идущих в выходные дни

Console.WriteLine("Фильмы, идущие в выходные дни (Суббота и Воскресенье):\n");

foreach (var poster in loadedPosters)

{

if (poster.SessionDate.DayOfWeek == DayOfWeek.Saturday || poster.SessionDate.DayOfWeek == DayOfWeek.Sunday)

{

poster.DisplayInfo();

}

}

// Подсчет количества киносеансов и их средней длительности для заданной даты

Console.WriteLine("Введите дату для подсчета количества киносеансов и их средней длительности (yyyy-MM-dd):");

string dateInput = Console.ReadLine();

DateTime inputDate = DateTime.ParseExact(dateInput, "yyyy-MM-dd", CultureInfo.InvariantCulture);

CalculateSessionsAndAverageDuration(loadedPosters, inputDate);

// Поиск первого сеанса для заданного кинотеатра и даты

Console.WriteLine("Введите название кинотеатра:");

string cinemaName = Console.ReadLine();

Console.WriteLine("Введите дату для поиска первого сеанса (yyyy-MM-dd):");

dateInput = Console.ReadLine();

inputDate = DateTime.ParseExact(dateInput, "yyyy-MM-dd", CultureInfo.InvariantCulture);

FindFirstSession(loadedPosters, cinemaName, inputDate);

// Подготовка справочной информации о киносеансах на следующий месяц

PrepareMonthlyReport(loadedPosters);

Console.ReadKey();

}

// Метод для записи массива киноафиш в файл

static void WriteToFile(CinemaPoster[] posters, string filePath)

{

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (var poster in posters)

{

writer.WriteLine(poster.ToString());

}

}

}

// Метод для чтения массива киноафиш из файла

static CinemaPoster[] ReadFromFile(string filePath)

{

List<CinemaPoster> posters = new List<CinemaPoster>();

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

posters.Add(CinemaPoster.FromString(line));

}

}

return posters.ToArray();

}

// Метод для подсчета количества сеансов и их средней длительности для заданной даты

static void CalculateSessionsAndAverageDuration(CinemaPoster[] posters, DateTime date)

{

var sessions = posters.Where(p => p.SessionDate.Date == date.Date).ToArray();

int count = sessions.Length;

double averageDuration = count > 0 ? sessions.Average(p => p.DurationMinutes) : 0;

Console.WriteLine($"На {date.ToShortDateString()} количество киносеансов: {count}");

Console.WriteLine($"Средняя длительность киносеансов: {averageDuration} минут");

}

// Метод для поиска первого сеанса для заданного кинотеатра и даты

static void FindFirstSession(CinemaPoster[] posters, string cinemaName, DateTime date)

{

var session = posters

.Where(p => p.CinemaName.Equals(cinemaName, StringComparison.OrdinalIgnoreCase) && p.SessionDate.Date == date.Date)

.OrderBy(p => p.StartTime)

.FirstOrDefault();

if (session.Equals(default(CinemaPoster)))

{

Console.WriteLine($"Сеансы не найдены для {cinemaName} на {date.ToShortDateString()}");

}

else

{

Console.WriteLine("Первый сеанс:");

session.DisplayInfo();

}

}

// Метод для подготовки справочной информации о киносеансах на следующий месяц

static void PrepareMonthlyReport(CinemaPoster[] posters)

{

var nextMonth = DateTime.Now.AddMonths(1).Month;

var nextMonthPosters = posters.Where(p => p.SessionDate.Month == nextMonth);

foreach (var group in nextMonthPosters.GroupBy(p => p.CinemaName))

{

string cinemaFileName = $"{group.Key}.txt";

using (StreamWriter writer = new StreamWriter(cinemaFileName))

{

foreach (var poster in group)

{

writer.WriteLine(poster.ToString());

}

}

Console.WriteLine($"Справочная информация о киносеансах для {group.Key} сохранена в файл {cinemaFileName}");

}

}

}

**Вывод:** изучил набор функций, предназначенных для работы с данными типа дата и время.