**Лабораторная работа №1.**

Требуется написать консольное приложение, выполняющее некоторые вычисления, согласно выбранного варианта. Приложение должно быть отказоустойчивым, то есть нельзя допустить аварийного завершения работы приложения. Все входные данные передаются в приложение с помощью некоторого количества входных файлов. Имена входных файлов передаются в программу как аргументы командной строки. Результат работы приложения также записывается в файл. По результатам выполнения работы необходимо написать отчет.

1. Информационная система поликлиники. Пользователь может просмотреть расписание врачей и записаться на приём, после чего должно быть сформировано подтверждение записи.

**Виды записи:**

В поликлинике №3 есть возможность для пациента просмотреть расписание врачей и записаться на прием

1. Онлайн;
2. по телефону;
3. через стойку регистратуры;
4. по знакомству через врача;
5. по направлению от других врачей (например, из другой больницы)

**Список врачей:**

В данной поликлинике присутствуют такие врачи как:

1. Терапевт
2. Хирург
3. Невролог
4. Окулист либо офтальмолог
5. Травматолог
6. Эндокринолог
7. Стоматолог

**Особенности**

Так же существует разделение на детских врачей и врачей для взрослых.

Записаться к некоторым врачам можно только по направлению от терапевта.

У каждого пациента присутствует медицинская карточка. Если медицинская карта заранее находилась у врача, то для некоторых врачей прием станет быстрее, из-за возможность просмотреть историю болезни. При любом виде записи, если мед карта находилась в регистратуре, то до приема ее заносят к врачу. В конце рабочего дня врачи относят мед карты обследованных пациентов на регистратуру. Если пациент в один день проходит несколько врачей, то он имеет возможность забирать ее сам.

В кабинетах некоторых врачей, периодически происходит кварцевание.

У каждого врача существует обеденный перерыв.

**Процесс записи**

Если запись происходит онлайн, то пациент сначала выбирает тип врача (детский или взрослый), после чего предоставляется список врачей, и после выбора нужного всплывает список услуг и их цены. После выбора нужной услуги, отображается график работы необходимых врачей и возможность записаться.

На регистратуре пациент может забрать или положить мед карту. Процесс самой записи идентичен с записью онлайн или по телефону.

Если пациент лично знаком с врачом, то этот врач сам выделяет ему место в графике. Запись в таком случае происходит в любом виде личного контакта.

Если пациент получает направление от врача этой поликлиники, то этот же врач имеет возможность его записать к нужному и спросить время у пациента. Если направление из другой поликлиники, то пациент имеет возможность записаться только через регистратуру, с предоставлением этого направления.

Врачи имеют несколько квалификаций. Существуют узкие и широкие специалисты.

Привязка кабинета к специалисту.

Ограниченное время приема (30 минут) + пауза (5 минут).

Формирование цены.

**Чтение/запись файлов**

Для чтения данных из файлов в C# применяются классы FileStream, StreamReader, StreamWriter, BinaryReader, BinaryWriter. При работе с данными классами необходимо самостоятельно освобождать выделенные ресурсы, поэтому при создании экземпляров данных классов нужно использовать конструкцию using или try-catch-finally с вызовом метода Close() в блоке finally.

Класс FileStream позволяет считывать и записывать данные как в бинарные, так и в текстовые файлы. Основными свойствами и методами данного класса являются:

* Свойство Length: возвращает длину потока в байтах
* Свойство Position: возвращает текущую позицию в потоке
* Метод Read: считывает данные из файла в массив байтов. Принимает три параметра: int Read(byte[] array, int offset, int count) и возвращает количество успешно считанных байтов. Здесь используются следующие параметры:
  + array – массив байтов, куда будут помещены считываемые из файла данные
  + offset – смещение в байтах в массиве array, в который считанные байты будут помещены
  + count – максимальное число байтов, предназначенных для чтения. Если в файле находится меньшее количество байтов, то все они будут считаны.
* Метод long Seek(long offset, SeekOrigin origin): устанавливает позицию в потоке.
  + offset – задает смещение в байтах от позиции origin.
  + origin – позиция, к которой метод будет применять смещение offset. SeekOrigin.Begin –начало файла, SeekOrigin.End – конец файла, SeekOrigin.Current – текущая позиция в файле.
* Метод Write: записывает в файл данные из массива байтов. Принимает три параметра: Write(byte[] array, int offset, int count)
  + array – исходный массив байтов, откуда данные будут записываться в файл
  + offset - смещение в байтах в массиве array, с которого начинается запись байтов в поток
  + count - максимальное число байтов, предназначенных для записи

Классы StreamReader и StreamWriter предназначены соответственно для чтения и записи текстовых файлов. Основные методы класса StreamReader:

* Peek – возвращает следующий доступный символ, если символов больше нет, то возвращает -1
* Read – считывает и возвращает следующий символ. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов
* ReadLine: считывает одну строку в файле
* ReadToEnd: считывает весь текст из файла

Основные методы класса StreamWriter

* Flush – записывает в файл оставшиеся в буфере данные и очищает буфер.
* Write – записывает в файл данные простейших типов, как int, double, char, string и т.д.
* WriteLine – записывает данные, и после записи добавляет в файл символ окончания строки

При работе с данными классами следует обращать внимание на используемую кодировку.

**Аргументы командной строки.**

Аргументы командной строки используются для передачи некоторых данных в приложение при запуске программы. Чаще всего через аргументы командной строки передаются настройки дальнейшей работы приложения. Аргументы командной строки являются параметрами метода Main:

void Main(string[] args);

Для того, чтобы задать аргументы командной строки при отладке приложения в Visual Studio зайдите в свойства проекта, выберите пункт «Отладка» и через пробел введите необходимое количество параметров в поле «Аргументы командной строки»

**Работа с Консолью**

Для очистки консоли можно использовать метод Console.Clear();

Проверить нажата какая-либо кнопка или нет можно используя метод KeyAvailable класса Console. Данный метод не блокирует выполнение программы, в отличие от методов Console.ReadKey() и Console.ReadLine():

/// <summary>

/// Метод показывает блокирующий способ обработки нажатия клавиши

/// </summary>

private static void BlockingRead()

{

while (true)

{

//Выполнение программы продолжается только после нажатия клавиши

Console.WriteLine("Выполнение...");

ConsoleKeyInfo cki = Console.ReadKey(true);

if (cki.Key == ConsoleKey.Escape)

{

Console.WriteLine("Выполнение прервано");

break;

}

else

{

Console.WriteLine($"Нажата клавиша {cki.Key}");

Console.WriteLine("Выполнение...");

}

}

}

/// <summary>

/// Метод показывает неблокирующий способ обработки нажатия клавиши

/// </summary>

private static void NonBlockingRead()

{

while (true)

{

//Выполнение программы продолжается не ожидая нажатия клавиши

if (Console.KeyAvailable == false)

{

Console.WriteLine("Выполнение...");

}

else

{

ConsoleKeyInfo cki = Console.ReadKey(true);

if (cki.Key == ConsoleKey.Escape)

{

Console.WriteLine("Выполнение прервано");

break;

}

else

{

Console.WriteLine($"Нажата клавиша {cki.Key}");

Console.WriteLine("Выполнение...");

}

}

}

}

**Метрики кода.**

Для расчета метрик кода в Visual Studio выберите пункт меню «Анализ –> Вычислить метрики кода» и нужный проект или все решение. Приведенные метрики необходимо привести в отчете. Число строк кода подсчитывается самостоятельно, а не берется из предоставляемых метрик.

**Содержание отчета.**

* Титульный лист с правильными годом, названием предмета, фамилиями, должностью преподавателя;
* Лист задания;
* Теоретическая часть. Анализ задания и обоснование метода решения. Описание используемых методов и алгоритмов, терминов (кроме общеизвестных) (На этом этапе еще нет ваших классов и методов);
* Практическая часть. Описание созданной объектной модели (включая описание методов, свойств, полей созданных классов), выбранных методов представления данных, сложностей, возникших при реализации задания варианта;
* Результаты метрик кода с выделением наиболее объемных методов, классов и т.д.
* Выводы по проделанной работе;
* Список использованной литературы;
* Полный текст программы в приложении;