3Лабораторная работа №8
Работа с файлами

1. Цель работы:

1) Получение практических навыков при работе со структурами.
2) Получение практических навыков при создании диалоговых Windows-приложений.
3) Получение практических навыков при работе с файлами.

2. Теоретические сведения

2.1. Понятие потока

Под **файлом** обычно подразумевается именованная информация на внешнем носителе, например на жестком или гибком магнитном диске. Логически файл можно представить как конечное количество последовательных байтов, поэтому такие устройства, как дисплей, клавиатура и принтер, также можно рассматривать как частные случаи файлов.

Обмен данными реализуется с помощью **потоков**. Поток (stream) — это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику.

Поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер).

Обмен с потоком для повышения скорости передачи данных производится, как правило, через специальную область оперативной памяти — **буфер**. Буфер выделяется для каждого открытого файла. При записи в файл вся информация сначала направляется в буфер и там накапливается до тех пор, пока весь буфер не заполнится. Только после этого или после специальной команды сброса происходит передача данных на внешнее устройство. При чтении из файла данные вначале считываются в буфер, причем не столько, сколько запрашивается, а сколько помещается в буфер.

Для поддержки потоков библиотека .NET содержит иерархию классов.

Основные классы пространства имен System.IO

Класс	Описание
BinaryReader, BinaryWriter	Чтение и запись значений простых встроенных типов (целочисленных, логических, строковых и т. п.) во внутренней форме представления
FileStream	Произвольный (прямой) доступ к файлу, представленному как поток байтов
StreamWriter, StreamReader	Чтение из файла и запись в файл текстовой информации (произвольный доступ не поддерживается)

Таким образом, выполнять обмен с внешними устройствами можно на уровне:

- двоичного представления данных (BinaryReader, BinaryWriter);
- байтов (FileStream);
- текста, то есть символов (StreamWriter, StreamReader).

В .NET используется кодировка Unicode, в которой каждый символ кодируется двумя байтами. Классы, работающие с **текстом**, являются оболочками классов, использующих байты, и автоматически выполняют перекодирование из байтов в символы и обратно.

Двоичные и байтовые потоки хранят данные в том же виде, в котором они представлены в оперативной памяти, то есть при обмене с файлом происходит побитовое

копирование информации. Двоичные файлы применяются не для просмотра их человеком, а для использования в программах.

Доступ к файлам может быть **последовательным**, когда очередной элемент можно прочитать (записать) только после аналогичной операции с предыдущим элементом, и **прямым**, при котором выполняется чтение (запись) произвольного элемента по заданному адресу. Текстовые файлы позволяют выполнять только последовательный доступ, в двоичных и байтовых потоках можно использовать оба метода.

Прямой доступ в сочетании с отсутствием преобразований обеспечивает высокую скорость получения нужной информации.

Использование классов файловых потоков в программе предполагает следующие операции:

1. Создание потока и связывание его с физическим файлом.
2. Обмен (ввод-вывод).
3. Закрытие файла.

Каждый класс файловых потоков содержит несколько вариантов конструкторов, с помощью которых можно создавать объекты этих классов различными способами и в различных режимах. Например, файлы можно открывать только для чтения, только для записи или для чтения и записи.

Режимы доступа к файлу содержатся в перечислении FileAccess, определенном в пространстве имен System.IO.

Значение	Описание
Read	Открыть файл только для чтения
ReadWrite	Открыть файл для чтения и записи
Write	Открыть файл только для записи

Возможные режимы открытия файла определены в перечислении FileMode

Значение	Описание
Append	Открыть файл, если он существует, и установить текущий указатель в конец файла. Если файл не существует, создать новый файл.
Create	Создать новый файл. Если в каталоге уже существует файл с таким же именем, он будет стерт.
CreateNew	Создать новый файл. Если в каталоге уже существует файл с таким же именем, возникает исключение IOException.
Open	Открыть существующий файл.
OpenOrCreate	Открыть файл, если он существует. Если нет, создать файл с таким именем
Truncate	Открыть существующий файл. После открытия он должен быть обрезан до нулевой длины.

2.2. Исключительные ситуации при работе с файлами

Операции по открытию файлов могут завершиться неудачно, например, при ошибке в имени существующего файла или при отсутствии свободного места на диске, поэтому рекомендуется всегда контролировать результаты этих операций.

В случае непредвиденных ситуаций среда выполнения генерирует различные исключения, обработку которых следует предусмотреть в программе, например:

- FileNotFoundException, если файла с указанным именем в указанном каталоге не существует;
- DirectoryNotFoundException, если не существует указанный каталог;

- Argument Except ion, если неверно задан режим открытия файла;
- IOException, если файл не открывается из-за ошибок ввода-вывода.

При закрытии файла освобождаются все связанные с ним ресурсы, например, для файла, открытого для записи, в файл выгружается содержимое буфера. Поэтому рекомендуется всегда закрывать файлы после окончания работы, в особенности файлы, открытые для записи. Если буфер требуется выгрузить, не закрывая файл, используется метод Flush.

2.3. FileStream (Потоки байтов)

Ввод-вывод в файл на уровне байтов выполняется с помощью класса FileStream, который является наследником абстрактного класса Stream, определяющего набор стандартных операций с потоками.

Элементы класса Stream

Значение	Описание
BeginRead, BeginWrite	Начать асинхронный ввод или вывод
CanRead, CanSeek, CanWrite	Свойства, определяющие, какие операции поддерживает поток: чтение, прямой доступ и/или запись
Close	Закрыть текущий поток и освободить связанные с ним ресурсы (сокеты, указатели на файлы и т. п.)
EndRead, EndWrite	Ожидать завершения асинхронного ввода; закончить асинхронный вывод
Flush	Записать данные из буфера в связанный с потоком источник данных и очистить буфер. Если для данного потока буфер не используется, то этот метод ничего не делает
Length	Возвратить длину потока в байтах
Position	Возвратить текущую позицию в потоке
Read, ReadByte	Считать последовательность байтов (или один байт) из текущего потока и переместить указатель в потоке на количество считанных байтов
Seek	Установить текущий указатель потока на заданную позицию
SetLength	Установить длину текущего потока
Write, WriteByte	Записать последовательность байтов (или один байт) в текущий потоки переместить указатель в потоке на количество записанных байтов

Текущая позиция в потоке первоначально устанавливается на начало файла (для любого режима открытия, кроме Append) и сдвигается на одну позицию при записи каждого байта.

Для установки желаемой позиции чтения используется метод Seek, имеющий два параметра: первый задает смещение в байтах относительно точки отсчета, задаваемой вторым. Точки отсчета задаются константами перечисления SeekOrigin:

- начало файла — Begin,
- текущая позиция — Current,
- конец файла — End.

2.4. StreamWriter и StreamReader (Потоки символов)

Символьные потоки StreamWriter и StreamReader работают с Unicode-символами, следовательно, ими удобнее всего пользоваться для работы с файлами, предназначенными для восприятия человеком. Эти потоки являются наследниками классов TextWriter и TextReader соответственно, которые обеспечивают их большей частью функциональности.

КлассTextWriter

Значение	Описание
Close	Закрыть файл и освободить связанные с ним ресурсы. Если в процессе записи используется буфер, он будет автоматически очищен
Fl ush	Очистить все буферы для текущего файла и записать накопленные в них данные в место их постоянного хранения. Сам файл при этом не закрывается
NewLine	Используется для задания последовательности символов, означающих начало новой строки.
Write	Записать фрагмент текста в поток
WriteLine	Записать строку в поток и перейти на другую строку

Класс TextReader

Значение	Описание
Peek	Возвратить следующий символ, не изменяя позицию указателя в файле
Read	Считать данные из входного потока
ReadBlock	Считать из входного потока указанное пользователем количество символов и записать их в буфер, начиная с заданной позиции
ReadLine	Считать строку из текущего потока и возвратить ее как значение типа string. Пустая строка (null) означает конец файла (EOF)
ReadToEnd	Считать все символы до конца потока, начиная с текущей позиции, и возвратить считанные данные как одну строку типа string

2.5. Двоичные файлы (BinaryWriter, BinaryReader)

Двоичные файлы (BinaryWriter, BinaryReader) хранят данные в том же виде, в котором они представлены в оперативной памяти, то есть во внутренней форме представления. Двоичные файлы применяются не для просмотра их человеком, а для использования в программах.

Выходной поток BinaryWriter поддерживает произвольный доступ, то есть имеется возможность выполнять запись в произвольную позицию двоичного файла. Двоичный файл открывается на основе базового потока, в качестве которого чаще всего используется поток FileStream. Входной двоичный поток содержит перегруженные методы чтения для всех простых встроенных типов данных.

Основные методы двоичных потоков.

Класс BinaryWriter

Значение	Описание

BaseStream	Базовый поток, с которым работает объект BinaryWriter
Close	Закрыть поток
Fl ush	Очистить все буферы для текущего файла и записать накопленные в них данные в место их постоянного хранения. Сам файл при этом не закрывается
Seek	Установить позицию в текущем потоке
Write	Записать фрагмент текста в поток

Класс BinaryReader

Значение	Описание
BaseStream	Базовый поток, с которым работает объект BinaryWriter
Close	Закрыть поток
Peek Char	Возвратить следующий символ, не изменяя позицию указателя в файле
Read	Считать данные из входного потока и сохранить в массиве, передаваемом как входной параметр
ReadXXXX	Считать из входного потока данные определенного типа

2.7. Понятие структуры и класса

Класс – это модуль, архитектурная единица построения программной системы. Модульность построения – основное свойство программных систем. Объектно-ориентированная программная система, строящаяся по модульному принципу, состоит из классов, являющихся основным видом модуля. Размер и содержание модуля определяется архитектурными соображениями, а не семантическими. Модульность построения – основное средство борьбы со сложностью системы.

Класс – это тип данных, задающий реализацию некоторой абстракции данных, характерной для проблемной области, в интересах которой создается программная система. В общем случае класс содержит **данные**, задающие свойства объектов класса, и **функции**, определяющие их поведение. В класс также могут добавляться **события**, на которые может реагировать объект класса (приложения, построенные на основе событийно-управляемой модели, например, при программировании для Windows.).

Упрощенное описание класса.

class имя_класса тело-класса

Данные, содержащиеся в классе, могут быть переменными (поля) или константами. Некоторые спецификаторы полей и констант класса.

- public - доступ к элементу не ограничен.
- protected - доступ только из данного и производных классов.
- private - доступ только из данного класса.
- static - одно поле для всех экземпляров класса

По умолчанию элементы класса считаются закрытыми (private). Все методы класса имеют непосредственный доступ к его закрытым полям.

Метод (функция) — реализует вычисления или другие действия, выполняемые классом или экземпляром. Методы определяют поведение класса.

Синтаксис метода:

```
[ спецификаторы ] тип имя_метода ( [ параметры ] ) тело метода
```

Чаще всего для методов задается спецификатор доступа public, ведь методы составляют интерфейс класса - то, с чем работает пользователь, поэтому они должны быть доступны .

Статические (static) методы, или методы класса, можно вызывать, не создавая экземпляр объекта. Именно таким образом используется метод Main.

Для каждого **объекта** при его создании в памяти выделяется отдельная область, в которой хранятся его данные. Объекты создаются явным или неявным образом, то есть либо программистом, либо системой. Программист создает экземпляр класса с помощью операции new. При этом для создания объекта и инициализации его данных вызывается специальный метод – конструктор. Имя конструктора совпадает с именем класса.

Свойства конструкторов:

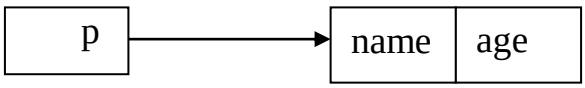
1. Конструктор не возвращает значение, даже типа `void`.
2. Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации.
3. Если программист не указал ни одного конструктора или какие-то поля не были инициализированы, полям значимых типов присваивается нуль, полям ссылочных типов – значение null .

Классы - это ссылочные типы, т. е. доступ к объектам классов осуществляется по адресу (через ссылку). Доступ к объектам классов через ссылки увеличивает расходы системных ресурсов, в том числе и памяти.

```
Person p1 = new Person();
p1.Show();
Person p2 = new Person("Иванов", 21);
p2.Show();

p1 = p2;
p2.name = "Петров";

p1.Show();
p2.Show();
```



Структура подобна классу, но она является значимым, а не ссылочным типом.

Описание структуры.

struct имя_структуры тело_структуры

Структуры могут содержать:

- методы,
- поля,
- конструкторы **с параметрами**,
- события.

Конструктор по умолчанию автоматически определяется для всех структур, и его изменить нельзя.

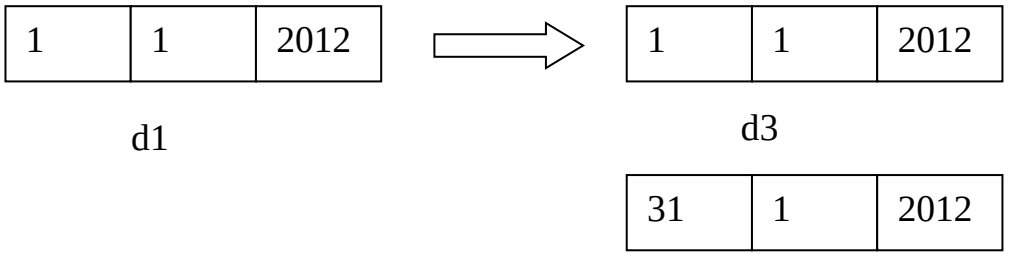
```
struct Date
    {
        public int year, month, day;
        public  Date(int d, int m, int y)
        {
            year = y; month = m; day = d;
        }
        public override string ToString()
```

```
        {
            string result;
            if (day < 10) result = "0" + day.ToString() + ":";
            else result = day.ToString() + ":";
            if (month < 10) result = result + "0" +
month.ToString() + ":";
            else result = result + month.ToString() + ":";
            result = result + year.ToString();
            return result;
        }
        public void Show()
        {
            Console.WriteLine(this.ToString());
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Date d1=new Date (1,1,2012);
            Console.WriteLine(d1.ToString());
            Date d2=new Date();
            d2.Show();
//создали без использования new, но поля непроинициализированы
          Date d3;
          //d3.Show();//Error;
            d3.day = 5;
            d3.month = 10;
            d3.year = 2013;
            d3.Show();
        }
    }
```

Объекты типа Структура можно создавать с помощью операции new (также как и классы), в этом случае поля инициализируются конструктором. Если объект создается без использования конструктора (d3 в примере 1), то поля останутся не проинициализированы и им нужно будет присвоить значения вручную.

При присваивании одной структуры другой создается копия этого объекта.

```
d3 = d1;//копия
d3.day = 31;
Console.Write("d3= "); d3.Show();
Console.Write("d1= "); d1.Show();
```



2.8. Сериализация

В C# есть возможность сохранять на внешних носителях не только данные примитивных типов, но и объекты. Сохранение объектов называется сериализацией, а восстановление

сохраненных объектов — десериализацией. При сериализации объект преобразуется в линейную последовательность байтов. Это сложный процесс, поскольку объект может включать множество унаследованных полей и ссылки на вложенные объекты, которые, в свою очередь, тоже могут состоять из объектов сложной структуры.
К счастью, сериализация выполняется автоматически, достаточно просто пометить класс как сериализуемый с помощью атрибута [Serializable]. Атрибуты — это дополнительные сведения о классе, которые сохраняются в его метаданных. Те поля, которые сохранять не требуется, помечаются атрибутом [NonSerialized].

Объекты можно сохранять в одном из двух форматов: двоичном или SOAP (в виде XML-файла). В первом случае следует подключить к программе пространство имен System.Runtime.Serialization.Formatters.Binary, во втором — пространство System.Runtime.Serialization.Formatters.Soap.

Рассмотрим *сохранение объектов в двоичном формате.* Для этого используется класс BinaryFormatter, в котором определены два метода:

- Serialize(поток, объект) - сохраняет заданный объект в заданном потоке;

Deserialize (поток) - восстанавливает объект из заданного потока.

Для *сохранения объекта в двоичном формате необходимо:*

1. Подключить к программе пространство имен System.Runtime.Serialization.

Formatters.Binary.

2. Пометить сохраняемый класс и связанные с ним классы атрибутом [Serializable].
3. Создать поток и связать его с файлом на диске или с областью оперативной памяти.
4. Создать объект класса BinaryFormatter.
5. Сохранить объекты в потоке.
6. Закрыть файл.

Пример.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace ConsoleApplication5
{
    [Serializable]
    struct Person
    {
     public   string name;
        protected int age;
        [NonSerialized]
        DateTime date;//дата создания объекта
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        public int Age
        {
            get { return age; }
            set { if (value > 0 && value < 99) age = value; else age = 0; }
        }
        public DateTime Date
        {
            get { return date; }
```

```
            set { date = DateTime.Now; }
        }
        public Person()
        {
            name = ""; age = 0; date = DateTime.Now;
        }
        public Person(string N, int A)
        {
            name = N; age = A; date = DateTime.Now;
        }

       virtual public void Show()
        {
            Console.WriteLine(Name + ", " + Age + ", объект создан " + date);
        }

       virtual public void Input()
       {
           Console.Write("Имя:"); name = Console.ReadLine();
           Console.Write("Возраст"); age = Convert.ToInt32(Console.ReadLine());

       }


    }
    [Serializable]
    class Student : Person
    {
        protected int kurs;
        protected double rating;

        public int Kurs
        {
            set { kurs = value; }
            get { return kurs; }
        }
        public double Rating
        {
            set { rating = value; }
            get { return rating; }

        }
        public Student()
            : base()
        {
            rating = 0; kurs = 0;
        }
        public Student(string N, int A, int K, double R)
            : base(N, A)
        {
            kurs = K; rating = R;
        }

        public override void Show()
        {
            base.Show();
            Console.WriteLine("курс " + kurs + ", рейтинг " + rating);

        }
        public override void Input()
        {
            base.Input();
            Console.Write("курс"); kurs = Convert.ToInt32(Console.ReadLine());
            Console.Write("Рейтинг"); rating = Convert.ToDouble(Console.ReadLine());
        }
```

```
    }


    class Program
    {
        static void Main(string[] args)
        {
            Person p1 = new Person();
            p1.Show();
            Person p2 = new Person("Иванов", 20);
            p2.Show();
            p1.Input();
            p1.Show();

            Student s=new Student("Петров",20,2,4.0);
            s.Show();

            FileStream f1 = new FileStream("Demo.bin", FileMode.Create );
            BinaryFormatter bf = new BinaryFormatter();
            bf.Serialize( f1, p1 ); // сохранение объекта  в потоке f
            bf.Serialize( f1, p2 ); // сохранение объекта  в потоке f
            bf.Serialize(f1, s);
            f1.Close();
             Console.WriteLine("Объекты сохранены в поток");
             Console.WriteLine("Выгружаем объекты из потока");
            FileStream f2 = new FileStream( "Demo.bin", FileMode.Open );

            Person d = (Person) bf.Deserialize(f2 ); // восстановление объекта
            d.Show();

            Person x = (Person)bf.Deserialize(f2);
            x.Show();
            Student y = (Student)bf.Deserialize(f2);
            y.Show();

            f2.Close();
        }
    }
}
```

3. Постановка задачи

Написать Windows-приложение для работы с простой базой данных, хранящей информацию об объекте на внешнем носителе. Приложение должно выполнять следующие функции:

1. Создание базы данных, содержащей записи указанного формата.
2. Просмотр базы данных.
3. Удаление элементов из базы данных (по ключу/ по номеру).
4. Корректировка элементов в базе данных (по ключу / по номеру).
5. Добавление элементов в базу данных (в начало / в конец/ с заданным номером).
6. Выполнение задания, указанного в варианте.

4. Методические указания

7. Для выбора действий использовать меню.
8. Для выбора файлов использовать стандартные диалоговые окна.
9. Для добавления элементов в файл использовать вспомогательный файл. Добавление должно осуществляться в начало, в конец файла и на позицию с заданным номером.

10. Удаление должно осуществляться по ключевому полю и по номеру. Удаляемые из файла записи помечаются как удаленные, когда их количество превысит половину файла, их можно удалять из файла с помощью вспомогательного файла.
11. Корректировка выполняется по ключу и по номеру.

5. Содержание отчета

1. Постановка задачи (общая и для конкретного варианта).
2. Анализ задачи.
3. Проектирование функций для реализации поставленных задач (название функции, назначение, входные параметры, результат).
4. Тесты для каждой функции.
5. Тест для комплексной задачи (интеграция).
6. Листинг программы.

6. Варианты

1. В типизированном файле хранится информация о средней температуре за месяц. Программа должна
   a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
   b. Определять:
      i. дни, в которые температура поднималась выше средней за месяц;
      ii. самый длинный отрезок между днями с отрицательной температурой.

2. Типизированный файл содержит данные о ежемесячных доходах подразделений фирмы «Феникс» за пять лет. Программа должна
   a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
   b. Определять:
      i. для каждого подразделения самые прибыльные годы;
      ii. наиболее длинный период каждого подразделения с доходом ниже среднего по всей фирме.

3. Списки студентов нескольких групп 1 курса хранятся в отдельных файлах – один файл на одну группу. Программа должна
   a. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
   b. Создавать новый файл, в котором будет храниться общий список студентов 1 курса, отсортированный по алфавиту.
   c. Создавать новый файл, в котором будет храниться общий список студентов 1 курса, отсортированный по убыванию номеров групп.

4. Типизированный файл, имеет следующую структуру: Автор, Название, Год издания, Издательство. Программа должна
   a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
   b. Выдавать по запросу пользователя:
      i. список литературы, указанного пользователем автора;
      ii. список литературы, изданной в указанный пользователем период.

5. Типизированный файл содержит данные о клиентах банка, получивших кредит (ФИО клиента, сумма кредита, вид кредита, срок, на который выдан кредит). Клиент может взять несколько кредитов.
   a. Программа должна

i. Добавлять, удалять, корректировать, позволять просматривать записи файлов.

b. Выполнять:

i. вывод списка клиентов, получивших указанный пользователем вид кредита;

ii. определение клиентов, взявших самый большой суммарный кредит.

6. В типизированном файле хранится информация о средней температуре за месяц. Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.

b. Определять:

i. среднюю температуру каждой недели;

ii. самые тёплые и самые холодные дни;

.

7. Типизированный файл содержит данные о ежемесячных доходах подразделений фирмы «Феникс» за пять лет. Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.

b. Определять:

i. для каждого подразделения самые прибыльные годы;

ii. наиболее длинный период каждого подразделения с доходом ниже среднего по всей фирме.

8. Списки студентов нескольких групп 1 курса хранятся в отдельных файлах – один файл на одну группу. Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файлов.

b. Создавать новый файл, в котором будет храниться общий список студентов 1 курса, отсортированный по убыванию номеров групп.

9. Типизированный файл, имеет следующую структуру: Автор, Название, Год издания, Издательство. Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.

b. Выдавать по запросу пользователя:

i. список литературы, указанного пользователем автора;

ii. список литературы, изданной в указанный пользователем период.

10. Типизированный файл содержит данные о клиентах банка, получивших кредит (ФИО клиента, сумма кредита, вид кредита, срок, на который выдан кредит). Клиент может взять несколько кредитов.

a. Программа должна

i. Добавлять, удалять, корректировать, позволять просматривать записи файлов.

b. Выполнять:

i. вывод списка клиентов, получивших указанный пользователем вид кредита;

ii. определение клиентов, взявших самый большой суммарный кредит.

11. В типизированном файле хранится информация о средней температуре за месяц. Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.

b. Определять:

i. среднюю температуру каждой недели;
ii. самый длинный отрезок между днями с отрицательной температурой.

12. Типизированный файл содержит данные о ежемесячных доходах подразделений фирмы «Феникс» за пять лет. Программа должна
a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Определять:
i. средний доход каждого подразделения за пять лет;
ii. наиболее длинный период каждого подразделения с доходом ниже среднего по всей фирме.

13. Списки студентов нескольких групп 1 курса хранятся в отдельных файлах – один файл на одну группу. Программа должна
a. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Создавать новый файл, в котором будет храниться общий список студентов 1 курса, отсортированный по алфавиту.

14. Типизированный файл, имеет следующую структуру: Автор, Название, Год издания, Издательство. Программа должна
a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Выдавать по запросу пользователя:
i. список имеющейся литературы;
ii. список литературы, изданной в указанный пользователем период.

15. Типизированный файл содержит данные о клиентах банка, получивших кредит (ФИО клиента, сумма кредита, вид кредита, срок, на который выдан кредит). Клиент может взять несколько кредитов.
a. Программа должна
i. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Выполнять:
i. поиск клиентов, взявших кредит на N месяцев;
ii. поиск клиентов по шаблону (3 первые буквы фамилии);

16. В типизированном файле хранится информация об автобусах, находящихся в автобусном парке (номер маршрута, ФИО водителя, номер автобуса, состояние автобуса (в парке или на маршруте)). Программа должна
a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Определять:
i. количество автобусов на маршруте;
ii. формировать список водителей автобусов, находящихся в парке.

17. Типизированный файл содержит данные о сотрудниках фирмы «Феникс» (ФИО, адрес, телефон, образование, название подразделения, в котором работает сотрудник). Программа должна
a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Определять:
i. количество сотрудников с высшим образованием в каждом подразделении;

ii. список сотрудников для заданного подразделения.

18. Типизированный файл содержит данные о студентах 1 курса (ФИО, группа, адрес, дата рождения, итоговые оценки по дисциплинам). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Выводить список студентов, имеющих хотя бы одну задолженность.
c. Выводить список предметов, по которым есть хотя бы одна задолженность.

19. Типизированный файл хранит информацию о товарах в магазине (наименование, цена, дата поступления, срок годности товара, название отдела, в котором хранится товар). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Выдавать по запросу пользователя:
    i. список просроченных продуктов в заданном отделе.
    ii. суммарную стоимость продуктов указанного наименования.

20. Типизированный файл содержит данные о клиентах банка, получивших кредит (ФИО клиента, сумма кредита, вид кредита, срок, на который выдан кредит). Клиент может взять несколько кредитов.

a. Программа должна
    i. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Выполнять:
    i. вывод списка клиентов, получивших указанный пользователем вид кредита;
    ii. вывод фамилии клиента, взявшего самый большой суммарный кредит.

21. В типизированном файле хранится информация об автобусах, находящихся в автобусном парке (номер маршрута, ФИО водителя, номер автобуса, состояние автобуса (в парке или на маршруте)). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Выводить:
    i. информацию о водителе по шаблону (2 первые буквы фамилии);
    ii. список водителей автобусов, находящихся на маршруте.

.

22. Типизированный файл содержит данные о сотрудниках фирмы «Феникс» (ФИО, адрес, телефон, образование, название подразделения, в котором работает сотрудник). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Выводить:
    i. информацию о заданном сотруднике по шаблону (первые 3 буквы фамилии);
    ii. фамилии сотрудников с высшим образованием для заданного подразделения.

23. Типизированный файл содержит данные о студентах 1 курса (ФИО, группа, адрес, дата рождения, итоговые оценки по дисциплинам). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Выводить:

i. список студентов, не имеющих задолженностей.
ii. средний балл за сессию заданного студента.

24. Типизированный файл хранит информацию о товарах в магазине (наименование, цена, дата поступления, срок годности товара, название отдела, в котором хранится товар). Программа должна

a. Добавлять, удалять, корректировать, позволять просматривать записи файла.
b. Выдавать по запросу пользователя:
    i. информацию о продуктах, поступивших в определенную дату.
    ii. список продуктов в заданном отделе.

25. Типизированный файл содержит данные о машинах, находящихся в гараже (ФИО владельца, сумма платежа в месяц, внесена ли плата за текущий месяц, марка машины, модель машины, год выпуска, номер машины, оплаченная сумма). У владельца может быть несколько машин.

a. Программа должна добавлять, удалять, корректировать, позволять просматривать записи файлов.
b. Выполнять:
    i. формирование списка к оплате за текущий месяц;
    ii. поиск машины по номеру;