

**СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ УО
«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ А.А. КУЛЕШОВА»**



Дисциплина

«Конструирование программ и языки программирования»

Разработка программ с использованием структур и перечислений

(2 часа)

Методические рекомендации к лабораторно работе №9

Могилев

Понятие «Структуры», «Перечисления». Методические указания по лабораторной работе №9 по дисциплине «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2-40 01 01 «Программное обеспечение информационных технологий».

ОГЛАВЛЕНИЕ

1. Цель работы.....	4
2. Краткие теоретические сведения.....	5
3. Задания.....	12
4. Контрольные вопросы.....	16

1. Цель работы

Приобретение навыков практического применения, закрепления знаний при создании простейших программ с использованием структур и перечислений

Ход работы

1. Изучение теоретического материала.
2. Выполнение практических индивидуальных заданий по вариантам (вариант уточняйте у преподавателя).
3. Оформление отчета.
 - 3.1. Отчет оформляется индивидуально каждым студентом. Отчет должен содержать задание, алгоритм и листинг программы.
 - 3.2. Отчет по лабораторной работе выполняется на листах формата А4. В состав отчета входят:
 1. титульный лист;
 2. цель работы;
 3. текст индивидуального задания;
 4. выполнение индивидуального задания.
4. Контрольные вопросы.

2 Краткие теоретические сведения

2.1 Общие сведения о структурах

Известно, классы относятся к ссылочным типам данных. Это означает, что объекты конкретного класса доступны по ссылке, в отличие от значений простых типов, доступных непосредственно. Но иногда прямой доступ к объектам как к значениям простых типов оказывается полезно иметь, например, ради повышения эффективности программы. Ведь каждый доступ к объектам (даже самым мелким) по ссылке связан с дополнительными издержками на расход вычислительных ресурсов и оперативной памяти.

Для разрешения подобных затруднений в C# предусмотрена структура, которая подобна классу, но относится к типу значения, а не к ссылочному типу данных. Т.е. структуры отличаются от классов тем, как они сохраняются в памяти и как к ним осуществляется доступ (классы — это ссылочные типы, размещаемые в куче, структуры — типы значений, размещаемые в стеке), а также некоторыми свойствами (например, структуры не поддерживают наследование). Из соображений производительности вы будете использовать структуры для небольших типов данных. Однако в отношении синтаксиса структуры очень похожи на классы.

Главное отличие состоит в том, что при их объявлении используется ключевое слово `struct` вместо `class`. Ниже приведена общая форма объявления структуры:

```
struct имя : интерфейсы {  
    // объявления членов  
}
```

где `имя` обозначает конкретное имя структуры.

Как и у классов, у каждой структуры имеются свои члены: методы, поля, индексаторы, свойства, операторные методы и события. В структурах допускается также определять конструкторы, но не деструкторы. В то же время для структуры нельзя определить конструктор, используемый по умолчанию (т.е. конструктор без параметров). Дело в том, что конструктор, вызываемый по умолчанию, определяется для всех структур автоматически и не подлежит изменению. Такой конструктор инициализирует поля структуры значениями, задаваемыми по умолчанию. А поскольку структуры не поддерживают наследование, то их члены нельзя указывать как `abstract`, `virtual` или `protected`.

Объект структуры может быть создан с помощью оператора `new` таким же образом, как и объект класса, но в этом нет особой необходимости. Ведь когда используется оператор `new`, то вызывается конструктор, используемый по умолчанию. А когда этот оператор не используется, объект по-прежнему создается, хотя и не инициализируется. В этом случае инициализацию любых членов структуры придется выполнить вручную.

Рассмотрим пример использования структур:

```
using System;  
  
namespace ConsoleApplication1  
{  
    // Создадим структуру
```

```

struct UserInfo
{
    public string Name;
    public byte Age;

    public UserInfo(string Name, byte Age)
    {
        this.Name = Name;
        this.Age = Age;
    }

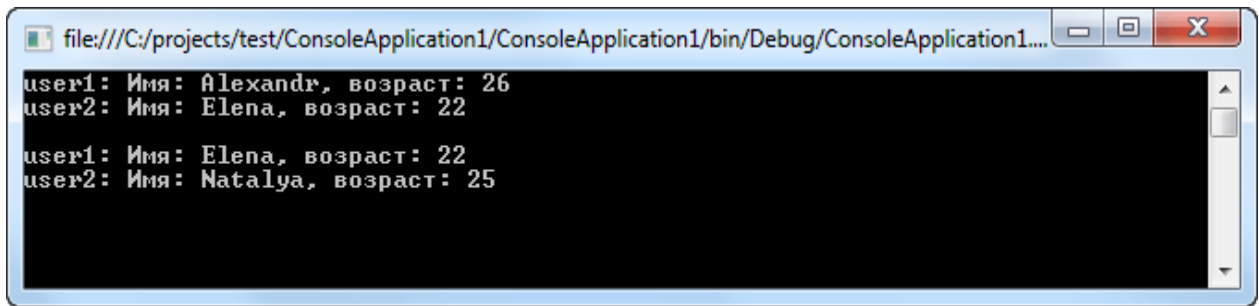
    public void WriteUserInfo()
    {
        Console.WriteLine("Имя: {0}, возраст: {1}", Name, Age);
    }
}

class Program
{
    static void Main()
    {
        UserInfo user1 = new UserInfo("Alexandr", 26);
        Console.Write("user1: ");
        user1.WriteUserInfo();
        UserInfo user2 = new UserInfo("Elena", 22);
        Console.Write("user2: ");
        user2.WriteUserInfo();

        // Показать главное отличие структур от классов
        user1 = user2;
        user2.Name = "Natalya";
        user2.Age = 25;
        Console.Write("\nuser1: ");
        user1.WriteUserInfo();
        Console.Write("user2: ");
        user2.WriteUserInfo();

        Console.ReadLine();
    }
}

```

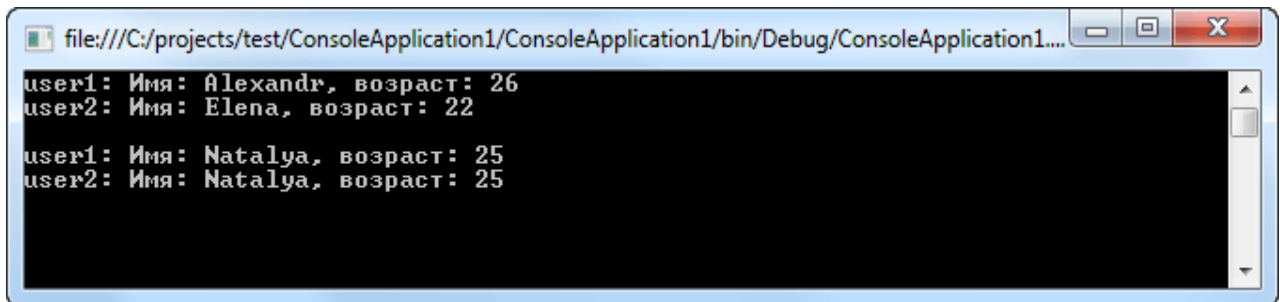


```
file:///C:/projects/test/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1....
user1: Имя: Alexandr, возраст: 26
user2: Имя: Elena, возраст: 22

user1: Имя: Elena, возраст: 22
user2: Имя: Natalya, возраст: 25
```

Обратите внимание, когда одна структура присваивается другой, создается копия ее объекта. В этом заключается одно из главных отличий структуры от класса. Когда ссылка на один класс присваивается ссылке на другой класс, в итоге ссылка в левой части оператора присваивания указывает на тот же самый объект, что и ссылка в правой его части. А когда переменная одной структуры присваивается переменной другой структуры, создается копия объекта структуры из правой части оператора присваивания.

Поэтому, если бы в предыдущем примере использовался класс `UserInfo` вместо структуры, получился бы следующий результат:



```
file:///C:/projects/test/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1....
user1: Имя: Alexandr, возраст: 26
user2: Имя: Elena, возраст: 22

user1: Имя: Natalya, возраст: 25
user2: Имя: Natalya, возраст: 25
```

2.1.1 Назначение структур

В связи с изложенным выше возникает резонный вопрос: зачем в `C#` включена структура, если она обладает более скромными возможностями, чем класс? Ответ на этот вопрос заключается в повышении эффективности и производительности программ. Структуры относятся к типам значений, и поэтому ими можно оперировать непосредственно, а не по ссылке. Следовательно, для работы со структурой вообще не требуется переменная ссылочного типа, а это означает в ряде случаев существенную экономию оперативной памяти.

Более того, работа со структурой не приводит к ухудшению производительности, столь характерному для обращения к объекту класса. Ведь доступ к структуре осуществляется непосредственно, а к объектам – по ссылке, поскольку классы относятся к данным ссылочного типа. Косвенный характер доступа к объектам подразумевает дополнительные издержки вычислительных ресурсов на каждый такой доступ, тогда как обращение к структурам не влечет за собой подобные издержки. И вообще, если нужно просто сохранить группу связанных вместе данных, не требующих наследования и обращения по ссылке, то с точки зрения производительности для них лучше выбрать структуру.

Любопытно, что в `C++` также имеются структуры и используется ключевое слово `struct`. Но эти структуры отличаются от тех, что имеются в `C#`. Так, в `C++` структура относится к типу класса, а значит, структура и класс в этом языке практически равноценны и отличаются друг от друга лишь доступом по умолчанию к их членам, которые оказываются закрытыми для класса и открытыми для структуры. А в `C#` структура относится к типу значения, тогда как класс – к ссылочному

типу.

2.2 Перечисления

Перечисления представляют набор логически связанных констант. Объявление перечисления происходит с помощью оператора `enum`. Далее идет название перечисления, после которого указывается тип перечисления - он обязательно должен представлять целочисленный тип (`byte`, `int`, `short`, `long`). Если тип явным образом не указан, то умолчанию используется тип `int`. Затем идет список элементов перечисления через запятую:

```
enum Days                                enum Time : byte
{ Monday,                                {
    Tuesday,                               Morning,
    Wednesday,                             Afternoon,
    Thursday,                               Evening,
    Friday,                                 Night
    Saturday,
    Sunday
}
```

В этих примерах каждому элементу перечисления присваивается целочисленное значение, причем первый элемент будет иметь значение 0, второй - 1 и так далее. Мы можем также явным образом указать значения элементов, либо указав значение первого элемента:

```
enum Operation
{
    Add = 1,    // каждый следующий элемент по умолчанию увеличивается на
единицу
    Subtract, // этот элемент равен 2
    Multiply, // равен 3
    Divide    // равен 4
}
```

Но можно и для всех элементов явным образом указать значения:

```
enum Operation
{
    Add = 2,
    Subtract = 4,
    Multiply = 8,
    Divide = 16
}
```

Каждое перечисление фактически определяет новый тип данных. Затем в программе мы можем определить переменную этого типа и использовать ее:


```

class Program
{
    enum Operation
    {
        Add = 1,
        Subtract,
        Multiply,
        Divide
    }
    static void Main(string[] args)
    {
        Operation op;
        op = Operation.Add;
        Console.WriteLine(op); // Add
        Console.ReadLine();
    }
}

```

В программе мы можем присвоить значение этой переменной. При этом в качестве ее значения должна выступать одна из констант, определенных в данном перечислении. То есть несмотря на то, что каждая константа сопоставляется с определенным числом, мы не можем присвоить ей числовое значение, например, `Operation op = 1;` И также если мы будем выводить на консоль значение этой переменной, то мы получим им константы, а не числовое значение. Если же необходимо получить числовое значение, то следует выполнить приведение к числовому типу:

```

Operation op;
op = Operation.Multiply;
Console.WriteLine((int)op); // 3

```

Также стоит отметить, что перечисление необязательно определять внутри класса, можно и вне класса, но в пределах пространства имен:

```

enum Operation
{
    Add = 1,
    Subtract,
    Multiply,
    Divide
}
class Program
{
    static void Main(string[] args)
    {
        Operation op;
        op = Operation.Add;
    }
}

```

```

        Console.WriteLine(op); // Add

        Console.ReadLine();
    }
}

```

Зачастую переменная перечисления выступает в качестве хранилища состояния, в зависимости от которого производятся некоторые действия. Так, рассмотрим применение перечисления на более реальном примере:

```

class Program
{
    enum Operation
    {
        Add = 1,
        Subtract,
        Multiply,
        Divide
    }

    static void MathOp(double x, double y, Operation op)
    {
        double result = 0.0;
        switch (op)
        {
            case Operation.Add:
                result = x + y;
                break;
            case Operation.Subtract:
                result = x - y;
                break;
            case Operation.Multiply:
                result = x * y;
                break;
            case Operation.Divide:
                result = x / y;
                break;
        }
        Console.WriteLine("Результат операции равен {0}", result);
    }

    static void Main(string[] args)
    {
        // Тип операции задаем с помощью константы Operation.Add, которая

```

равна 1

```
MathOp(10, 5, Operation.Add);
```

// Тип операции задаем с помощью константы `Operation.Multiply`, которая равна 3

```
MathOp(11, 5, Operation.Multiply);
```

```
Console.ReadLine();
```

```
}
```

```
}
```

Здесь у нас имеется перечисление `Operation`, которое представляет арифметические операции. Также у нас определен метод `MathOp`, который в качестве параметров принимает два числа и тип операции. В основном методе `Main` мы два раза вызываем процедуру `MathOp`, передав в нее два числа и тип операции.

3. Задания

1. Выполнить задания по вариантам.

Вариант 1

Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов).

Описать перечисление

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT (записи должны быть упорядочены по возрастанию номера группы);
- вывод на экран фамилий и номеров групп для всех студентов, включенных, в массив, если средний балл студента больше 6,0 (если таких студентов нет, вывести соответствующее сообщение).

Вариант 2

Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT (записи должны быть упорядочены по возрастанию среднего балла);
- вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 9 и 10 (если таких студентов нет, вывести соответствующее сообщение).

Вариант 3

Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT (записи должны быть упорядочены по алфавиту);
- вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку меньше 4 (если таких студентов нет, вывести соответствующее сообщение).

Вариант 4

Описать структуру с именем AEROFLOT, содержащую следующие поля:

- название пункта назначения рейса;
- номер рейса;
- тип самолета.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из семи элементов типа AEROFLOT (записи должны быть упорядочены по возрастанию номера рейса);
- вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры (если таких рейсов нет, вывести соответствующее сообщение).

Вариант 5

Описать структуру с именем AEROFLOT, содержащую следующие поля:

- название пункта назначения рейса;
- номер рейса;
- тип самолета.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из семи элементов типа AEROFLOT (записи должны быть размещены в алфавитном порядке по названиям пунктов назначения);
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры (если таких рейсов нет, вывести соответствующее сообщение).

Вариант 6

Описать структуру с именем WORKER, содержащую следующие поля:

- фамилия и инициалы работника;
- название занимаемой должности;
- год поступления на работу.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа WORKER (записи должны быть упорядочены по алфавиту);
- вывод на экран фамилий работников, стаж работы которых превышает значение, введенное с клавиатуры (если таких работников нет, вывести соответствующее сообщение).

Вариант 7

Описать структуру с именем TRAIN, содержащую следующие поля:

- название пункта назначения;
- номер поезда;
- время отправления.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть размещены в алфавитном порядке по названиям пунктов назначения);
- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких поездов нет, вывести соответствующее сообщение).

Вариант 8

Описать структуру с именем TRAIN, содержащую следующие поля:

- название пункта назначения;
- номер поезда;
- время отправления.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из шести элементов типа TRAIN (запи-

си должны быть упорядочены по времени отправления поезда);

- вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры (если таких поездов нет, вывести соответствующее сообщение).

Вариант 9

Описать структуру с именем TRAIN, содержащую следующие поля:

- название пункта назначения;
- номер поезда;
- время отправления.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть упорядочены по номерам поездов);

- вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких поездов нет, вывести соответствующее сообщение).

Вариант 10

Описать структуру с именем MARSH, содержащую следующие поля:

- название начального пункта маршрута;
- название конечного пункта маршрута;
- номер маршрута.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH (записи должны быть упорядочены по номерам маршрутов);

- вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких маршрутов нет, вывести соответствующее сообщение).

Вариант 11

Описать структуру с именем NOTE, содержащую следующие поля:

- фамилия, имя;
- номер телефона;
- дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE (записи должны быть упорядочены по дате рождения);

- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры (если такого нет, вывести соответствующее сообщение).

Вариант 12

Описать структуру с именем ZNAK, содержащую следующие поля:

- фамилия, имя;
- знак Зодиака;
- дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK (записи должны быть упорядочены по дате рождения);

- вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если такого нет, вы-

вести соответствующее сообщение).

Вариант 13

Описать структуру с именем PRICE, содержащую следующие поля:

- название товара;
- название магазина, в котором продается товар;
- стоимость товара в рублях.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE (записи должны быть упорядочены в алфавитном порядке по названиям товаров);
- вывод на экран информации о товаре, название которого введено с клавиатуры (если таких товаров нет, вывести соответствующее сообщение).

Вариант 14

Описать структуру с именем PRICE, содержащую следующие поля:

- название товара;
- название магазина, в котором продается товар;
- стоимость товара в рублях.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE (записи должны быть упорядочены в алфавитном порядке по названиям магазинов);
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры (если такого магазина нет, вывести соответствующее сообщение).

Вариант 15

Описать структуру с именем ORDER, содержащую следующие поля:

- расчетный счет плательщика;
- расчетный счет получателя;
- перечисляемая сумма в рублях.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ORDER (записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков);
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры (если такого расчетного счета нет, вывести соответствующее сообщение).

4 Контрольные вопросы

1. Что такое структуры? Чем их отличие от классов?
2. В чем назначение структур?
3. Что такое перечисления? Для каких целей они используются?