Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_\_\_\_Е.В.Фалей  « \_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2016 |

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная практика по программированию |
| Составлена в соответствии с учебной программой, утвержденной директором Колледжа бизнеса и права 31.05.2016 | |

**Практическая работа №9**

**Инструкционно-технологическая карта**

Тема: Перечисления. Структуры. Интерфейсы.

* Перечисления. Структуры (2 часа).
* Интерфейсы. Работа с объектами через интерфейсы (2 часа).
* Стандартные интерфейсы .NET (2 часа).

Цель: Закрепить умение работать с перечислениями и структурами. Развить навык работы с интерфейсами, с объектами через интерфейсы. Научиться применять стандартные интерфейсы .NET

Время выполнения: 6 часов

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Получить вариант индивидуального задания и выполнить его.
3. Оформить решенные задания в отчет.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

**Структуры**

*Структуры* - это составной объект, в который входят элементы любых типов, в том числе и функций. В отличие от массива, который является однородным объектом, структура может быть неоднородной, т.е. структура объединяет несколько переменных разного типа данных. Переменные, которые объ­единены структурой, называются членами, элементами или полями струк­туры.

Синтаксис структуры:

[ атрибуты ] [ спецификаторы ] struct имя\_структуры [ : интерфейсы ]

тело\_структуры [ ; ]

Из *спецификаторов доступа* допускаются только public и private (последний – только для вложенных структур).

*Тело структуры* может состоять из констант, полей, методов, свойств, событий, индексаторов, операций, конструкторов и вложенных типов.

Объявление структуры является оператором, и поэтому в конце должна стоять точка с запятой. *При этом никакая переменная не объявлена*. Выделения памяти под переменную не произошло.

При объявлении структуры задан так называемый *шаблон структуры* и определен новый пользовательский тип данных. Рассмотрим пример определения структуры студент. Обратите внимание на перегруженный метод ToString: он позволяет выводить экземпляры структуры (переменные типа структуры) на экран, поскольку неявно вызывается в методе Console.WriteLine.

namespace структура

{ //объявление структуры, обратите внимание на место в консольном приложении

public struct student

{

//поля структуры

public string name;

public int kurs;

public string gruppa;

public int stipendia;

// метод структуры (перегруженный)

public override string ToString()

{

return (string.Format( "Имя студента {0}; Курс{1}; Группа № {2};Размер стипендии: {3}" , name, kurs,gruppa,stipendia ) );

}//конец метода

}; //конец описания структуры student

class Program

{

static void Main(string[] args)

{

student s; //объявление экземпляра (переменной) структуры

Console.WriteLine("Введите данные о студенте:");

Console.WriteLine("Имя:");

//ДОСТУП к элементам структуры – через операцию “точка”

s.name = Console.ReadLine();

Console.WriteLine("Курс:");

s.kurs = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Группа");

s.gruppa = Console.ReadLine();

Console.WriteLine("Размер стипендии:");

s.stipendia = Convert.ToInt32(Console.ReadLine());

// вызов перегруженного метода ToString()

Console.WriteLine( "structura s: " +s); //вывод всех полей структуры на экран

Console.ReadKey();

}

}

}

При выводе экземпляра структуры на экран выполняется *упаковка*, то есть неявное преобразование в ссылочный тип. Упаковка применяется в других случаях, когда структурный тип используется там, где ожидается ссылочный, например, при преобразовании экземпляра структуры к типу реализуемого ею интерфейса. При обратном преобразовании – из ссылочного типа в структурный – выполняется *распаковка*.

*При присваивании структур* создается копия значений полей. То же самое происходит и при передаче структур в качестве параметров по значению. Для экономии ресурсов ничто не мешает передавать структуры в методы по ссылке с помощью ключевых слов ref или out.

Особенно значительный выигрыш в эффективности можно получить, используя массивы структур.

**Перечисления**

При написании программ часто возникает потребность определить несколько связанных между собой именованных констант, при этом их конкретные значения могут быть не важны. Для этого удобно воспользоваться перечисляемым типом данных, *все возможные значения которого задаются списком целочисленных констант,* например:

enum Menu { Read, Write, Append, Exit }

enum Радуга { Красный, Оранжевый, Желтый, Зеленый, Синий, Фиолетовый }

Для каждой константы задается ее символическое имя. По умолчанию константам присваиваются последовательные значения типа int, начиная с 0, но можно задать и собственные значения, например:

enum Nums { two = 2, three, four, ten = 10, eleven, fifty = ten + 40};

Константам three и four присваиваются значения 3 и 4, константе eleven – 11. Имена перечисляемых констант внутри каждого перечисления должны быть уникальными, а значения могут совпадать.

Преимущество перечисления перед описанием именованных констант состоит в том, что связанные константы нагляднее; кроме того, компилятор выполняет проверку типов, а интегрированная среда разработки подсказывает возможные значения констант, выводя их список.

**Операции с перечислениями**

С переменными перечисляемого типа можно выполнять:

* арифметические операции (+, –, ++, – –);
* логические поразрядные операции (^, &, |, ~);
* сравнить их с помощью операции отношения (<, <=, >, >=, = =, !=) ;
* получать размер в байтах (sizeof).

При использовании переменных перечисляемого типа в целочисленных выражениях и операциях присваивания требуется явное *преобразование типа*. Переменной перечисляемого типа можно присвоить *любое значение*, представимое с помощью базового типа, то есть не только одно из значений, входящих в тело перечисления. Присваиваемое значение становиться новым элементом перечисления. Рассмотрим пример.

struct Боец

{

public enum Воинское\_Звание

{

Рядовой, Сержант, Лейтенант, Майор, Полковник, Генерал

}

public string Фамилия;

public Воинское\_Звание Звание;

}

class Program

{

static void Main(string[] args)

{

Боец x;

x.Фамилия = "Иванов";

x.Звание = Боец.Воинское\_Звание.Сержант;

for (int i = 1976; i < 2006; i += 5)

{

if (x.Звание < Боец.Воинское\_Звание.Генерал) ++x.Звание;

Console.WriteLine("Год: {0} {1} {2}", i, x.Звание, x.Фамилия);

}

Console.ReadKey();

}

}

**Интерфейсы. Работа с объектами через интерфейсы**

Интерфейс – это контракт, обеспечивающий определенное поведение класса или структуры. Когда класс реализует интерфейс, он как бы говорит потенциальному пользователю: “Я гарантирую, что поддерживаю методы, свойства, события и индексаторы этого интерфейса”.

Интерфейс- семейство явно описанных как public – методов и свойств, которые сгруппированы в единое целое и инкапсулируют какую-либо определенную функциональную возможность.

После того как интерфейс определен, его можно реализовать в некотором классе. Это означает, что класс будет поддерживать все свойства и члены, определяемые данным интерфейсом. Интерфейсы не могут существовать сами по себе. таким же образом, как создается экземпляр класса. У интерфейса нет кода, который бы реализовал его члены; он просто описывает эти члены. Их реализация должна находиться в классах, в которых реализован данный интерфейс.

Итак, интерфейс представляет собой альтернативу абстрактному классу в смысле создания контрактов между классами и их пользователями. Эти контракты создаются с использованием ключевого слова interface, которое объявляет ссылочный тип, инкапсулирующий контракт. Синтаксически интерфейс подобен классу, имеющему только абстрактные методы. Абстрактный класс является базовым для семейства производных классов, в то время как интерфейсы задуманы для “смешивания” их с прочими деревьями наследования. Когда класс реализует интерфейс, он должен реализовать все методы этого интерфейса. По сути дела, он соглашается выполнить контракт, определенный интерфейсом.

Правила использования интерфейсов:

* нельзя "создать экземпляр интерфейса";
* интерфейс не может содержать в себе никакого кода;
* у класса может быть несколько интерфейсов, и несколько классов могут поддерживать один и тот же интерфейс;
* у интерфейсов нет никаких конструкторов;
* определению интерфейса не разрешено содержать перегрузки операций;
* не допускается указывать никаких модификаторов доступа, а также модификаторов virtual,static.

Рассмотрим интерфейсы за счет представления полного определения одного из интерфейсов, определенного Microsoft - System.IDisposable. Интерфейс IDisposable содержит один метод Dispose (), предназначенный для деинициализации классами, которые выполняют очистку кода:

public interface IDisposable

{

void Dispose ();

}

Интерфейс может содержать только объявления методов, свойств, индексов и событий. Он содержит только сигнатуры членов.

**Стандартные интерфейсы .NET**

В библиотеке .NET определено множество стандартных интерфейсов, задающих желаемое поведение объектов. Например, интерфейс IComparable задает метод сравнения объектов по принципу больше или меньше, что позволяет выполнять их сортировку. Реализация интерфейсов IEnumerable и IEnumerator дает возможность просматривать содержимое объекта с помощью конструкции foreach, а реализация интерфейса ICloneable – клонировать объекты.

Стандартные интерфейсы поддерживаются многими стандартными классами библиотеки. Например, работа с массивами с помощью цикла foreach возможна именно потому, что тип Array реализует интерфейсы IEnumerable и IEnumerator. Можно создавать и собственные классы, поддерживающие стандартные интерфейсы, что позволит использовать объекты этих классов стандартными способами.

Интерфейс IComparable содержит всего один метод **CompareTo(),** возвращающий результат сравнения двух объектов – текущего и переданного ему в качеству параметра:

interface IComparable

{

int CompareTo(object obj)

}

Метод должен возвращать:

* 0, если текущий объект и параметр равны;
* отрицательное число, если текущий объект меньше параметра;
* положительное число, если текущий объект больше параметра.

Рассмотрим пример использования стандартного интерфейса IComparable на примере класса Monster

namespace ConsoleApplication1

{

//создание класса, использующего интерфейс для сравнения элементов

class Monster : IComparable

{

public Monster(int сила, int умение, string имя)

{

this.сила = сила;

this.умение = умение;

this.имя = имя;

}

public void Passport()

{

Console.WriteLine("Монстр {0} \t сила = {1} умение= {2}", имя, сила, умение);

}

//обязательная реализация этого метода, т.к. он находится в интерфейсе

public int CompareTo(object obj)

{

Monster temp = (Monster)obj;

if (this.сила > temp.сила) return 1;

if (this.сила < temp.сила) return -1;

return 0;

}

string имя;

int сила, умение;

}

class Program

{

static void Main(string[] args)

{

const int n = 3;

Monster[] mas = new Monster[n]; //массив монстров

//инициализация каждого монстра(элемента массива)

mas[0] = new Monster(50, 50, "Вася");

mas[1] = new Monster(100, 100, "Петя");

mas[2] = new Monster(40, 10, "Игорь");

//Только теперь сортировка массива экземпляров класса Monster стала доступной, //благодаря методу CompareTo()

Array.Sort(mas);

//вывод элементов массива монстров

foreach (Monster x in mas)

x.Passport();

Console.ReadLine();

}

}

}

В данной программе критерием сортировки является поле сила для каждого монстра. То есть массив сортируется по возрастанию этого поля у всех экземпляров класса Monster.

Если несколько экземпляров имеют одинаковое значение критерия сортировки, то порядок их следования не изменится.

Во многих алгоритмах требуется выполнить сортировку по различным критериям. Для этого используется интерфейс IComparer.

Данный интерфейс определен в пространстве имен System.Collections. Он также содержит один метод Compare(), возвращающий результат сравнения двух объектов, переданных ему в качестве параметров.

interface IComparer

{

int Compare(object obj1,object obj2)

}

Принцип применения этого интерфейса состоит в том, что для каждого критерия сортировки объектов описывается небольшой вспомогательный класс, реализующий этот интерфейс. Объект этого класса передается в стандартный метод сортировки массива в качестве второго аргумента.

Модернизируем предыдущий пример и отсортируем массив экземпляров класса Monster по двум критериям – по именам (свойство Имя, класс SortByИмя) и умениям (свойство Умение, класс SortByУмение). Классы критериев сортировки объявлены вложенными, так как требуются только экземплярам класса Monster.

using System;

using System.Collections;//обязательное подключение данного пространства имен

namespace стандартинтерфейс

{

class Monster //обратите внимание на то, что класс не поддерживает интерфейс

{

public Monster(int сила, int умение, string имя)

{

this.сила = сила;

this.умение = умение;

this.имя = имя;

}

public int Умение

{

get { return умение; }

set

{

if (value > 0) умение = value;

else умение = 0;

}

}

public string Имя {

get { return имя; }

}

public void Passport()

{

Console.WriteLine("Монстр {0} сила = {1} умение= {2}", имя, сила, умение);

}

//Создание ВЛОЖЕННОГО КЛАССА для сортировки по одному критерию

public class SortByИмя : IComparer

{

//вызов стандартного метода Compare() интерфейса и реализация его для сортировки по //имени

int IComparer.Compare(object obj1, object obj2)

{

Monster m1 = (Monster)obj1;

Monster m2 = (Monster)obj2;

return String.Compare(m1.Имя, m2.Имя);

}

}

//Создание второго ВЛОЖЕННОГО КЛАССА для сортировки по одному критерию

public class SortByУмение : IComparer

{

//вызов стандартного метода Compare() интерфейса и реализация его для сортировки по //умению

int IComparer.Compare(object obj1, object obj2)

{

Monster m1 = (Monster)obj1;

Monster m2 = (Monster)obj2;

if (m1.Умение > m2.Умение) return 1;

if (m1.Умение < m2.Умение) return -1;

return 0;

}

}

string имя;

int сила, умение;

}

class Program

{

static void Main(string[] args)

{

const int n = 3;

Monster[] mas = new Monster[n];

//инициализация каждого монстра(элемента массива)

mas[0] = new Monster(50, 50, "Вася");

mas[1] = new Monster(100, 100, "Петя");

mas[2] = new Monster(40, 10, "Игорь");

Console.WriteLine("Сортировка по имени");

//метод Sort() вызывается с двумя параметрами!!!

//2-ой параметр – экземпляр класса Monster.SortByИмя – это полное имя класса!!!

Array.Sort(mas, new Monster.SortByИмя());

foreach (Monster x in mas)

x.Passport();

Console.WriteLine("Сортировка по умениям");

Array.Sort(mas, new Monster.SortByУмение());

foreach (Monster x in mas)

x.Passport();

Console.ReadLine();

}

}

}

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

**Общая постановка задачи**. Написать программу, реализующую алгоритм работы с заданной структурой. Продемонстрировать работу со стандартными интерфейсами .NET (например, сравнение объектов или их упорядочивание) и с перечислением.

Вариант 1

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 номера группы);

• вывод на экран фамилий и номеров групп для всех студентов, включенных, в массив, если средний балл студента больше 4,0 (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 йз десяти структур типа STUDENT (записи должны быть упорядочены по возрастанию среднего балла);

• вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (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 2 (если таких студентов нет, вывести соответствующее сообщение).

ВАРИАНТ 4

Описать структуру с именем AER0FL0T, 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 AER0FL0T (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 AER0FL0T, 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 AER0FL0T (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 (записи 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 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 маршрутах, которые начинаются или оканчиваются в пункте, название которого введено с клавиатуры (если таких маршрутов нет, вывести соответствующее сообщение).

ВАРИАНТ 12

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 13

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 размещены по алфавиту);

• вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет, вывести соответствующее сообщение).

ВАРИАНТ 14

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 NOT E (записи должны быть упорядочены по трем первым цифрам номера телефона);

• вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если такого нет, вывести соответствующее сообщение).

ВАРИАНТ 15

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, вывести соответствующее сообщение).

ВАРИАНТ 16

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 людях, родившихся под знаком, название которого введено с клавиатуры (если таких нет, вывести соответствующее сообщение).

ВАРИАНТ 17

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 знакам Зодиака);

• вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет, вывести соответствующее сообщение).

ВАРИАНТ 18

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 19

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 20

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).

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Албахари, Дж. C#5.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – 5-е изд. – М.: ООО «И.Д. Вильямс», 2014. – 1008 с.: ил.
2. Павловская Т.А. C#. Программирование на языке высокого уровня / Т.А. Павловская. – Учебник для вузов. – СПб.: Питер, 2014. – 432 с.: ил.
3. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд. – СПб.: Питер, 2014. – 816 с.: ил.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.
5. Шилдт, Г. C#4.0: Полное руководство / Г. Шилдт. – М.: ООО «И.Д. Вильямс», 2011. – 1056 с.: ил.
6. Шарп Джон. Microsoft Visual C#. Подробное руководство / Джон Шарп. – 8-е изд. – СПб.: Питер, 2017. – 848 с.
7. Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс] / Сайт о программировании. – metanit.com, 2012-2016. – Режим доступа: [http://metanit.com/sharp/ tutorial](http://metanit.com/sharp/%20tutorial). – Дата доступа: 20.08.2016.

Преподаватель Е.В. Багласова

|  |
| --- |
| Рассмотрено на заседании цикловой  Комиссии программного обеспечения информационных технологий № 10  Протокол №\_\_\_\_\_от «\_\_\_»\_\_\_\_2016  Председатель ЦК *( )* Т.Г. Багласова |