Лабораторная работа №9

Тема: Разработка программ с использованием структур и перечислений.

Цель: Научиться разрабатывать алгоритмы и реализовать программы с применением переменных структур и перечислений.

Теоретические сведения:

Перечисления

Перечисление (Enumeration) – это определяемый пользователем целочисленный тип, который позволяет специфицировать набор допустимых значений, и назначить каждому понятное имя.

Для объявления перечисления используется ключевое слово enum. Общая структура объявления перечисления выглядит так:

enum [имя_перечисления] { [имя1], [имя2], … };

Пример 1. Программа с использованием перечисления:

```
enum Directions { Left, Right, Forward, Back };
// объявление перечисления

class Program
{
  public static void GoTo(Directions direction)
  {
    switch (direction)
    {
      case Directions.Back:
        Console.WriteLine("Go back");
        break;
      case Directions.Forward:
        Console.WriteLine("Go forward");
        break;
      case Directions.Left:
        Console.WriteLine("Turn left");
        break;
      case Directions.Right:
        Console.WriteLine("Turn right ");
        break;
```

```
    }
  }

  static void Main(string[] args)
  {
    Directions direction = Directions.Forward;
    GoTo(direction); // "Go forward"
    Console.ReadKey();
  }
}
```

Чтобы получить целое значение определенного элемента перечисления, достаточно этот элемент явно привести к целому типу:

```
enum Directions : byte { Left, Right, Forward, Back };

class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine((int)Directions.Forward); // 2
    Console.ReadKey();
  }
}
```

По умолчанию в качестве целого типа для enum используется int. Этот тип можно изменить на любой другой целый тип (кроме char), указав после имени перечисления необходимый тип и разделив двоеточием:

```
enum Directions : byte { Left, Right, Forward, Back };
```

Зачем нужны перечисления и где они используются

Главные преимущества, которые нам дают перечисления это:

- Гарантия того, что переменным будут назначаться допустимые значения из указанного набора;

- Когда вы пишите код программы в Visual Studio, благодаря средству IntelliSense будет выпадать список с допустимыми значениями, что позволит сэкономить некоторое время, и напомнить, какие значения можно использовать;

- Код становится читабельнее, когда в нем присутствуют понятные имена, а не ни о чем не говорящие числа.

Перечисления очень широко используются в самой библиотеке классов .NET. Например, при создании файлового потока (FileStream) используется перечисление FileAccess, при помощи которого мы указываем с каким режимом доступа открыть файл (чтение/запись).

Следующий пример кода объявляет и использует тип enum с именем Color, в котором определены три константы: Red, Green, и Blue.

Пример 2:

```
using System;
enum Color
{
   Red,
   Green,
   Blue
}
class EnumExample
{
   static void PrintColor(Color color)
   {
      switch (color)
      {
         case Color.Red:
            Console.WriteLine("Red");
            break;
         case Color.Green:
            Console.WriteLine("Green");
            break;
         case Color.Blue:
            Console.WriteLine("Blue");
            break;
         default:
            Console.WriteLine("Unknown color");
            break;
      }
   }
   static void Main()
```

```
    {
        Color c = Color.Red;
        PrintColor(c);
        PrintColor(Color.Blue);
    }
}
```

Каждый тип enum соотносится с одними из целочисленных типов, который является *базовым типом* для этого типа enum. Если для типа enum базовый тип не объявлен явным образом, ему присваивается базовый тип int. Формат хранения и диапазон возможных значений для типа enum определяются его базовым типом. Набор значений, которые может принимать enum, не ограничивается его членами enum. В частности, любое значение базового типа enum можно привести к типу enum, и оно будет являться допустимым дискретным значением для этого типа enum.

Следующий пример кода объявляет тип enum с именем Alignment и базовым типом sbyte.

```
enum Alignment: sbyte
{
    Left = -1,
    Center = 0,
    Right = 1
}
```

Как показано в предыдущем примере, объявление члена enum может содержать константное выражение, задающее значение этого члена. Константное значение для каждого члена enum должно находиться в диапазоне, определенном для базового типа enum. Когда объявление члена enum не указывает значение явным образом, этому члену присваивается нулевое значение (если это первый элемент в типе enum) или значение, на единицу превышающее значение последнего объявленного члена enum.

Значения Enum можно преобразовать к целочисленным значениям, и наоборот, используя приведение типов. Пример:

```
int i = (int)Color.Blue;    // int i = 2;
Color c = (Color)2;         // Color c = Color.Blue;
```

Для любого типа enum по умолчанию устанавливается целочисленное нулевое значение, преобразованное к типу enum. В тех случаях, когда переменная автоматически инициализируются значением по умолчанию, переменным типов enum присваивается именно это значение. Чтобы

значение по умолчанию было легко доступно для любого типа enum, литеральное значение 0 неявным образом преобразуется к любому типу enum. Таким образом, допустимо следующее выражение.

```
Color c = 0;
```

Структуры

Объявляется структура со словом *struct*. Структура и класс очень похожи по функциональности. Структура как и класс содержит конструкторы, переменные, методы.

Пример описания структуры:

```
public struct Book
{
 // members
 public string Name;
 public int Price;

 // constructor
 public Book(string name, int price)
 {
 Name = name;
 Price = price;
 }

 // methods
 public void Show()
 {
 Console.WriteLine("Название книги {0}, Цена {1}", Name, Price);
 }
}
```

Объект структуры может быть создан 2-мя способами:

- без оператора new
- с оператором new

Пример 3. Создание структуры без оператор new.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
 public struct Book
 {
 // members
 public string Name;
 public int Price;

 // methods
 public void Show()
 {
 Console.WriteLine("Название книги {0}, Цена {1}", Name, Price);
 }
 }

 class Program
 {
 static void Main(string[] args)
 {
 // создаем структуру без оператора new
 Book book;

 // инициализируем данные класса
 book.Name = "Волшебник Земноморья";
 book.Price = 70;

 // Показываем результат на экране
 book.Show();
 }
 }
}
```

Пример 4. Создание структуры с оператором new, без параметров (по умолчанию).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
 public struct Book
 {
 // members
 public string Name;
 public int Price;

 // constructor
 public Book(String name, int price)
 {
 Name = name;
 Price = price;
 }

 // methods
 public void Show()
 {
 Console.WriteLine("Название книги {0}, Цена {1}", Name, Price);
 }
 }

 class Program
 {
 static void Main(string[] args)
 {
 // создаем объект структуры с оператором new без параметров
 Book book = new Book();
```

```
  // конструктор по умолчанию установит значения
  // Name = ""
  // Price = 0

  // Показываем результат на экране
  book.Show();
  }
  }
}
```

Примечание: Так создается объект структуры с оператором new без параметров.

Вызовется конструктор по умолчанию (конструктор без параметров)

Для структур мы не можем определить конструктор без параметров

```
public Book()
{
...
}
```

будет ошибка компиляции

Дело в том, что конструктор по умолчанию (конструктор без параметров), определяется для всех структур автоматически и его не видно в коде.

Пример 5. Создание структуры с оператором new, с параметрами.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
  public struct Book
  {
  // members
  public string Name;
  public int Price;
```

```
// constructor
public Book(String name, int price)
{
Name = name;
Price = price;
}

// methods
public void Show()
{
Console.WriteLine("Название книги {0}, Цена {1}", Name, Price);
}
}

class Program
{
static void Main(string[] args)
{
// создаем объект структуры с оператором new с параметрами
Book book = new Book("Волшебник Земноморья", 90);

// наш конструктор установит значения
// Name = "Волшебник Земноморья"
// Price = 90

// Показываем результат на экране
book.Show();
}
}
}
```

Выполнение работы:

Задание

Описать структуру согласно варианту задания. Написать функции, позволяющие:

1) вводить с клавиатуры данные;
2) реализовать запрос согласно варианту задания.

Вариант задания отдел кадров:
– ФИО работника;
– должность;
– дата рождения;
– ученая степень;
– стаж работы.
Запрос. Вывести на экран работников, стаж которых
превышает заданный. Вывести на экран работников выбранной должности.

Проектирование приложения

1. Запустите VS.

2. Создайте новый проект.

3. Добавьте компоненты:

textBox – ввод ФИО, стажа работника и данных для запроса,

label – формирование вспомогательных надписей,

groupBox – разделение группы элементов ввода данных и группы запросов к структуре,

menuStrip – меню программы,

dataGridView – таблицы для отображения данных структуры и результатов запроса,

radioButton – выбор варианта запроса,

comboBox – выбор должности и ученой степени из выпадающего списка,

button – кнопки для обработки событий,

dateTimePicker – календарь для выбора даты рождения.

По окончании проектирования форма примет вид, представленный на рисунках 1 – 2. Настройка всех элементов, кроме свойства Name и компонента меню, выполнена программно.

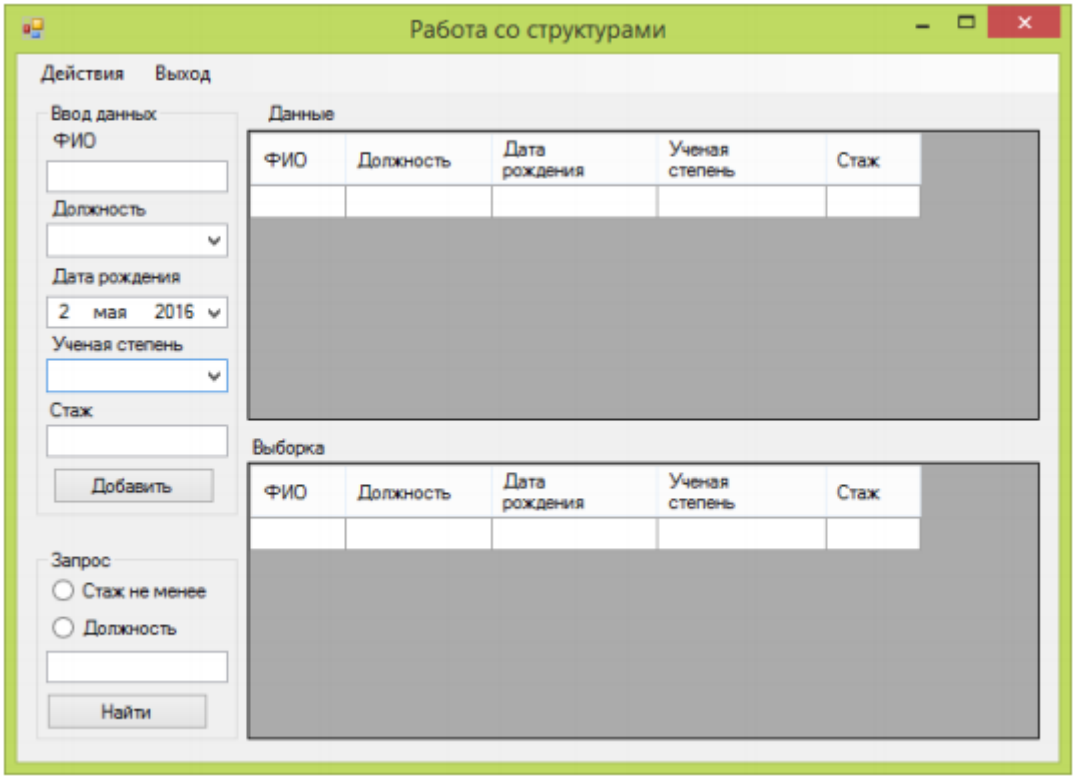

Рисунок 1 - Форма по окончании проектирования

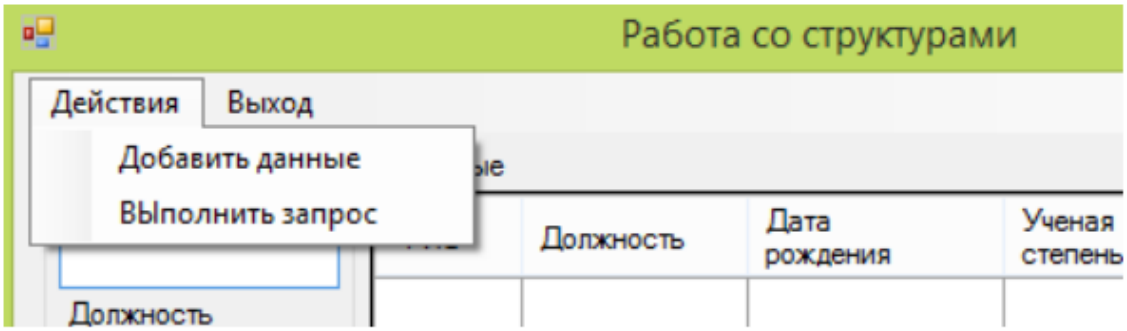

Рисунок 2 - Меню программы

4. Пункты меню *Добавить данные* и *Выполнить запрос* дублируют кнопки *Добавить* и *Найти* соответственно. По нажатию кнопки *Выход* происходит закрытие программы.

5. Создаем события обработки нажатия кнопок, пунктов меню и событие загрузки формы. Код файла Form1.cs представлен ниже.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Struct
{
struct Employee
{
public string FIO;//ФИО
public string Post;// Должность
public string Date_of_Birth;// Дата рождения
public string Degree; //Ученая степень
public int Experience;//Стаж работы
public Employee(string f, string p, string d, string
deg, int e)//конструктор
{
FIO = f;
Post = p;
Date_of_Birth = d;
Degree = deg;
Experience = e;
}
}
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
private void Form1_Load(object sender, EventArgs e)
{
comboBox1.Items.Add("Преподаватель");
comboBox1.Items.Add("Ст. преподаватель");
comboBox1.Items.Add("Доцент");
comboBox1.Items.Add("Профессор");
comboBox2.Items.AddRange(new object[] {"Без уч. степени","Кандидат наук",
"Доктор наук"});
dataGridView1.RowHeadersVisible = false;
```

```
dataGridView1.ColumnCount = 5;
dataGridView1.Columns[0].HeaderText = "ФИО";
dataGridView1.Columns[1].HeaderText = "Должность";
dataGridView1.Columns[2].HeaderText = "Дата рождения";
dataGridView1.Columns[3].HeaderText = "Ученая
степень";
dataGridView1.Columns[4].HeaderText = "Стаж";
dataGridView2.RowHeadersVisible = false;
dataGridView2.ColumnCount = 5;
dataGridView2.Columns[0].HeaderText = "ФИО";
dataGridView2.Columns[1].HeaderText = "Должность";
dataGridView2.Columns[2].HeaderText = "Дата рождения";
dataGridView2.Columns[3].HeaderText = "Ученая
степень";
dataGridView2.Columns[4].HeaderText = "Стаж";
dataGridView2.RowHeadersVisible = false;
}
Employee[] worker = new Employee[10];
int cout = 0;
private void button1_Click(object sender, EventArgs e)
{
worker[cout].FIO = textBox1.Text;
worker[cout].Post = comboBox1.Text;
worker[cout].Date_of_Birth =
dateTimePicker1.Value.ToString("dd.MM.yyyy");
worker[cout].Degree = comboBox2.Text;
worker[cout].Experience =
Convert.ToInt32(textBox2.Text);
dataGridView1.Rows.Add(worker[cout].FIO,
worker[cout].Post, worker[cout].Date_of_Birth,
worker[cout].Degree, worker[cout].Experience.ToString());
cout++;
}
private void button2_Click(object sender, EventArgs e)
{
if(radioButton1.Checked == true)
```

```
{
dataGridView2.Rows.Clear();
int select1 =
Convert.ToInt32(textBox3.Text);
foreach (Employee wSel in worker)
{
if (wSel.Experience >= select1)
dataGridView2.Rows.Add(wSel.FIO,
wSel.Post, wSel.Date_of_Birth, wSel.Degree,
wSel.Experience.ToString());
}
}
if (radioButton2.Checked == true)
{
dataGridView2.Rows.Clear();
string select2 = textBox3.Text;
foreach (Employee wSel in worker)
{
if (wSel.Post == select2)
dataGridView2.Rows.Add(wSel.FIO,
wSel.Post, wSel.Date_of_Birth, wSel.Degree, wSel.Experience.
ToString());
}
}
}
private void добавить ДанныеToolStrip MenuItem_Click
(object sender, EventArgs e)
{ button1_Click(sender, e); }
private void
вЫполнитьЗапросToolStripMenuItem_Click(object sender,
EventArgs e)
{ button2_Click(sender, e); }
private void выходToolStripMenuItem_Click(object
sender, EventArgs e)
{ Close(); }
```

Тестирование и использование приложения.

Пример выполнения приложения представлен на рисунках 3 – 4.

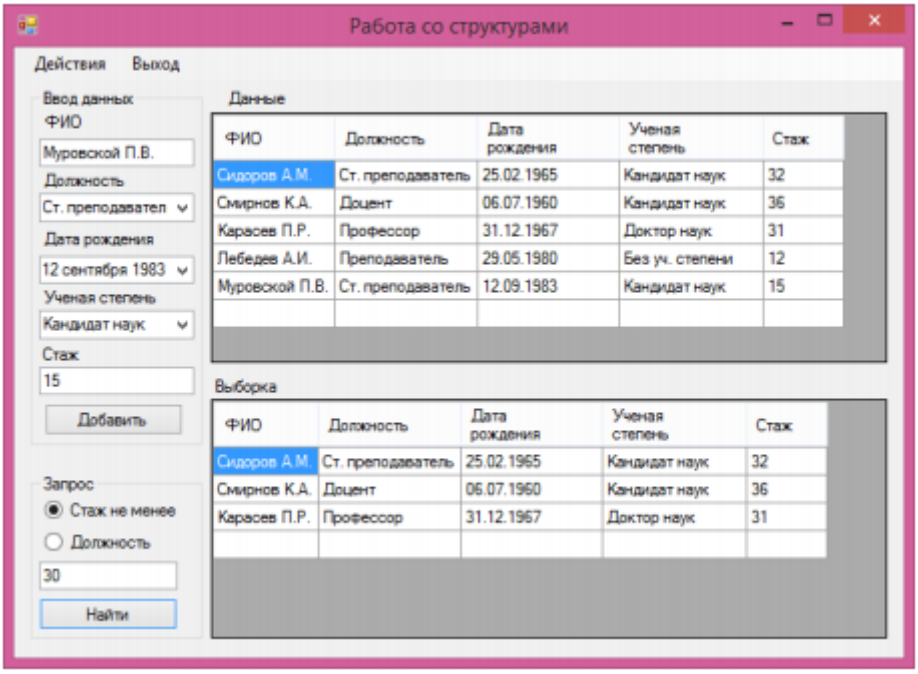

Рисунок 3 - Выборка по стажу

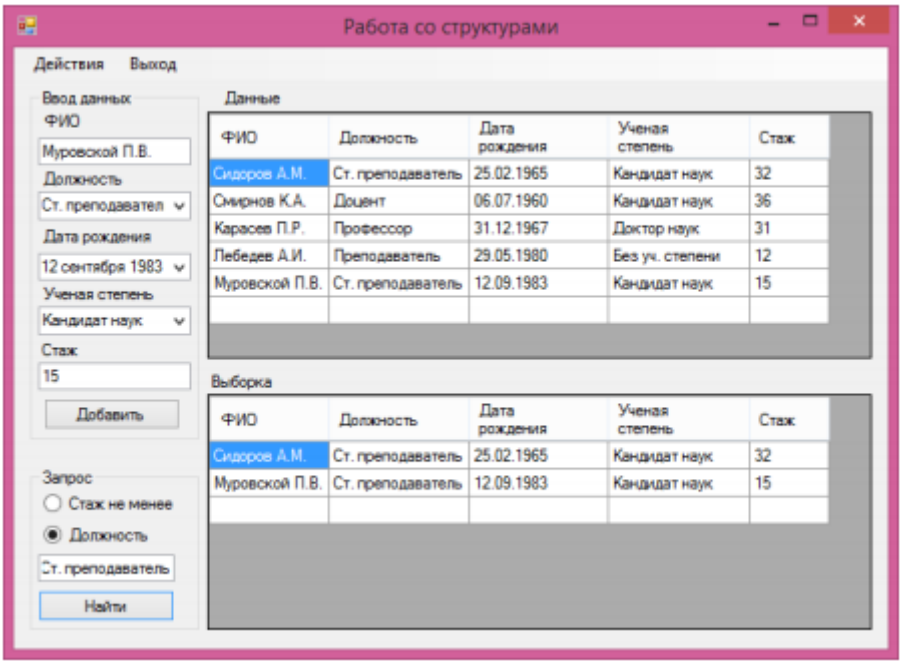

Рисунок 4 - Выборка по должности

1. Запустите приложение на выполнение.
2. Заполните структуры 4 – 5 наборами данных.
3. Выполните поиск сотрудников со стажем выше заданного порога.
4. Выполните поиск сотрудников соответствующей должности.

5. Нажмите кнопки меню, проверив их работоспособность.

Варианты индивидуальных заданий.

Вариант индивидуального задания получите у преподавателя.

1. Каталог книг:

• название;
• автор;
• количество страниц;
• год издания.

Запрос. Вывести на экран все книги данного автора.

2. Каталог газет:

• название газеты;
• номер;
• дата выхода;
• количество страниц.

Запрос. Вывести на экран все газеты, выходившие в определенном месяце.

3. Перечень факультетов:

• название факультета;
• ФИО декана;
• Телефон;
• адрес.

Запрос. Вывести на экран декана данного факультета.

4. Перечень кафедр:

• название кафедры;
• ФИО заведующего кафедрой;
• количество преподавателей;
• адрес.

Запрос. Вывести на экран кафедры, где количество преподавателей превышает заданное.

5. Перечень студентов:

• ФИО студента;
• дата рождения;
• адрес;
• телефон.

Запрос. Вывести на экран студентов, имеющих одинаковую дату рождения.

6. Рейтинг успеваемости студентов:

• ФИО студента;
• группа;
• средний бал;

• размер стипендии.

Запрос. Вывести на экран студентов, средний балл которых превышает заданный.

7. Перечень основных дисциплин:

• название дисциплины;
• кафедра, на которой читается дисциплина;
• ФИО преподавателя, читающего лекции;
• ФИО преподавателя, ведущего лабораторные занятия.

Запрос. Вывести на экран дисциплины, которые читаются преподавателями заданной кафедры.

8. Список дисциплин кафедры:

• Название;
• ФИО преподавателя;
• семестр, в котором читается дисциплина;
• группа.

Запрос. Вывести на экран дисциплины, отсортированные по преподавателям.

9. Расписание преподавателя:

• дата;
• день недели;
• предмет;
• группа;
• аудитория.

Запрос. Вывести на экран все пары преподавателя с заданной группой.

10. Список выданных книг:

• код книги;
• номер читательского билета;
• дата выдачи;
• срок выдачи.

Запрос. Вывести на экран книги, которые читатели не сдали вовремя в соответствии с заданныме числом.

11. Список товаров:

• наименования товара;
• единица измерения;
• количество на складе;
• цена за единицу.

Запрос. Рассчитать общую стоимость каждого товара на складе.

12.Список продаж:

• наименование товара;
• наименование покупателя;
• дата продажи;
• количество;
• стоимость.

Запрос. Вывести на экран суммарную стоимость покупок каждого покупателя.

13. Справочник улиц города:

• наименование улицы;
• длина;
• история;
• район.

Запрос. Вывести на экран улицы, отсортированные по районам.

14. Справочник пропусков студентов.

• ФИО студента;
• Группа;
• количество пропусков;
• количество неаттестаций.

Запрос. Вывести на экран студентов, количество пропусков которых превышает заданное.

Шкала оценивания индивидуальных заданий

Примеры	1-5 балла
Примеры + Индивидуальное задание (реализация программы через консоль)	1-8 баллов
Примеры + Индивидуальное задание (реализация программы с помощью формы)	1-10 баллов

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. Скриншоты выполнения примеров
5. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
6. Вывод по лабораторной работе