# Лабораторная работа № 19

**Тема:** Разработка программ на языке С# с использованием структур.

**Цель работы:**

1. Формирование навыков разработки программ на языке С# с использованием структур.
2. Изучение особенностей реализации структур на языке С#.

. **Оборудование:**

1. ПК
2. Программное обеспечение: ОС Windows, среда Visual Studio.Net

**Ответы на контрольные вопросы:**

1. В C# структура - это пользовательский тип данных, который может содержать поля и методы. Структуры используются для создания легких типов данных, которые обычно используются для представления простых объектов данных, например, координаты или информацию о дате и времени.

2. В C# структуры не наследуются от классов. Они являются типами значений (value types), а не ссылочными типами (reference types).

3. Для объявления структуры в C# используется ключевое слово "struct". Пример объявления структуры:

struct Point

{

public int X;

public int Y;

}

4. Для создания экземпляра структуры в C# используйте оператор "new" и вызов конструктора. Например, если у вас есть структура Point:

struct Point

{

public int X;

public int Y;

}

Вы можете создать экземпляр структуры следующим образом

Point p = new Point();

Вы также можете инициализировать поля структуры при создании экземпляра, указав значения в фигурных скобках:

Point p = new Point { X = 2, Y = 5 };

В обоих случаях будет создан новый экземпляр структуры Point, и вы сможете обращаться к его полям (X и Y).

5. Структуры не поддерживают наследование друг от друга или от классов. Они могут только реализовывать интерфейсы.

6. В структурах нельзя использовать ключевые слова "abstract", "virtual" или "protected". Они предназначены для классов.

7. Основное отличие между структурами и классами состоит в том, что структуры - это типы значений, которые хранятся непосредственно в стеке, в то время как объекты классов - это ссылочные типы, которые хранятся в куче. Когда экземпляр структуры присваивается другой переменной или передается в метод, происходит копирование значений, а не создание ссылки. Классы поддерживают наследование и имеют более широкие возможности для работы с объектами и состоянием.

**Ход работы:**

1. В соответствии с вариантом задания разработать алгоритм задачи и представить его в виде  программы на С#
2. Используя средства Visual Studio.Net создать файл с программой и выполнить тестирование и отладку программы
3. Результаты представить в виде отчета
4. Сделать вывод о проделанной работе.

**Варианты заданий для лабораторной работы №19**

Создайте структуру с именем Draftee, содержащую поля: фамилия, имя, отчество призывника, год его рождения, адрес, семейное положение, дату прохождения медкомиссии, заключение о пригодности к службе. Ввести данные в массив из пяти элементов типа Draftee, упорядочить элементы в алфавитном порядке по фамилии призывника. Добавить возможность вывода фамилий призывников заданного года рождения и заключением «Годен».

**Листинг с исходным кодом**

using System;

struct Draftee

{

public string Фамилия;

public string Имя;

public string Отчество;

public int ГодРождения;

public string Адрес;

public string СемейноеПоложение;

public string ДатаМедкомиссии;

public string Заключение;

public Draftee(string фамилия, string имя, string отчество, int годРождения, string адрес, string семейноеПоложение, string датаМедкомиссии, string заключение)

{

Фамилия = фамилия;

Имя = имя;

Отчество = отчество;

ГодРождения = годРождения;

Адрес = адрес;

СемейноеПоложение = семейноеПоложение;

ДатаМедкомиссии = датаМедкомиссии;

Заключение = заключение;

}

public override string ToString()

{

return Фамилия + " " + Имя + " " + Отчество; // Представление объекта в виде строки

}

}

class Program

{

static void Main(string[] args)

{

Draftee[] draft = new Draftee[5]; // Создание массива структур Draftee

for (int i = 0; i < draft.Length; i++) // Цикл для заполнения данных для каждого призывника

{

Console.WriteLine($"Введите данные для призывника №{i + 1}");

Console.Write("Фамилия: ");

string фамилия = Console.ReadLine();

Console.Write("Имя: ");

string имя = Console.ReadLine();

Console.Write("Отчество: ");

string отчество = Console.ReadLine();

Console.Write("Год рождения: ");

int годРождения = Convert.ToInt32(Console.ReadLine());

Console.Write("Адрес: ");

string адрес = Console.ReadLine();

Console.Write("Семейное положение: ");

string семейноеПоложение = Console.ReadLine();

Console.Write("Дата медкомиссии: ");

string датаМедкомиссии = Console.ReadLine();

Console.Write("Заключение: ");

string заключение = Console.ReadLine();

draft[i] = new Draftee(фамилия, имя, отчество, годРождения, адрес, семейноеПоложение, датаМедкомиссии, заключение); // Создание нового объекта Draftee с помощью конструктора и запись его в массив

Console.WriteLine();

}

Array.Sort(draft, (x, y) => x.Фамилия.CompareTo(y.Фамилия)); // Упорядочивание элементов массива по фамилии в алфавитном порядке

Console.WriteLine("Упорядоченные по фамилии призывники:");

foreach (Draftee draftee in draft) // Перебор и вывод призывников в отсортированном порядке

{

Console.WriteLine(draftee);

}

Console.WriteLine();

Console.Write("Введите год рождения: ");

int year = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите заключение: ");

string conclusion = Console.ReadLine();

Console.WriteLine($"Призывники, родившиеся в {year} году и имеющие заключение \"{conclusion}\":");

foreach (Draftee draftee in draft) // Перебор и вывод призывников, соответствующих введенным условиям

{

if (draftee.ГодРождения == year && draftee.Заключение == conclusion)

{

Console.WriteLine(draftee);

}

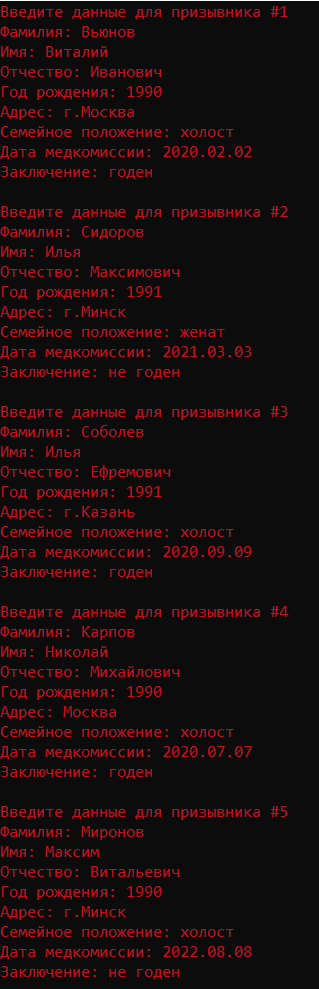
}

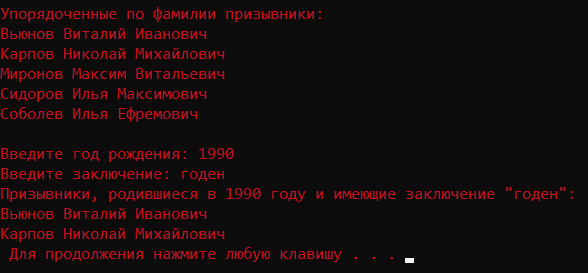
Console.ReadKey();

}

}

**Результаты тестирования**





**Вывод по работе.** При разработке программ с использованием структур необходимо учитывать особенности их использования и выбирать подходящий тип данных (класс или структуру) в зависимости от условий задачи и требований к производительности. Умение работать со структурами и разумно выбирать между классами и структурами поможет создать эффективные и надежные программы на языке C#.