Министерство науки и ВЫСШЕГО образования   
Российской Федерации

Федеральное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский государственный политехнический   
университет Петра Великого»  
**Институт среднего профессионального образования**

**Лабораторная работа № 4  
Тема:** Создание программ, использующих структуры и коллекции.

МДК.01.01 Разработка программных модулей

**Вариант 9**

Выполнил: студент группы 32919/8  
Крупина Елена

Проверил: преподаватель   
Прокофьев А.А.

Санкт-Петербург  
2024

**Цель работы:** изучить принципы описания и использования структур и коллекций

в языке C#, получить практические навыки разработки программ по обработке

коллекций и структур.

**Разработать следующие функции:**

* добавление в список структур новой записи;
* удаление из списка элемента с заданным индексом;
* вывод на экран всего списка структур;
* корректировки отдельной записи, найденной по содержимому главного
* поля;
* отбор записей по содержимому одного из полей и вывод на экран
* найденных записей;
* основное задание по вашему варианту.

**Особые требования:**

* Хотя бы одно из полей структуры должно иметь перечисляемый тип;
* Интерфейс программы оформить в виде меню.

**ВАРИАНТ 9**

Создать тип структура, имеющую следующие поля: номер группы

(перечисляемый тип), фамилия имя отчество студента, год рождения, телефон.

Найти в списке студентов определенной группы, не достигших

совершеннолетия. Найти в списке студентов, фамилия которых начинается на

заданную букву.

**Математическая формулировка:** Необходимо разработать программу, которой будет реализован цикл меню, из которого можно будет выйти, если пользователь введет определенное значение и функции по заданию, вызов которых будет реализован в меню.

**Блок-схема (функции проверки ввода)**









**Блок-схема (функции основного задания)**

















**Блок-схема (функции по варианту)**





**Блок-схема (функция Main()):**





**Код программы:**

using System.Globalization;

using System.Linq;

using System.Text.RegularExpressions;

class Program

{

static string Input(string message)

{

Console.Write(message);

string input = Console.ReadLine();

string inValid = "0123456789\_/:'\*!@%$:;";

for (int i = 0; i < inValid.Length; i++)

{

if (input.Contains(inValid[i]) || string.IsNullOrWhiteSpace(input))

{

Console.WriteLine("Некорректный ввод.");

Console.WriteLine();

return Input(message); // вызов метода для повторного ввода

}

}

// Проверка на корректность ввода

return input;

}

static Group InputGroup(string message)

{

Console.WriteLine(message);

string input = Console.ReadLine();

// Проверка на числовой формат

if (int.TryParse(input, out int num))

{

Console.WriteLine("Некорректное значение. Допустимые значения: G1, G2, G3, G4.");

Console.WriteLine();

return InputGroup(message);

}

// Проверка на корректность ввода

if (Enum.TryParse<Group>(input, true, out Group result))

{

return result;

}

else

{

Console.WriteLine("Некорректное значение. Допустимые значения: G1, G2, G3, G4.");

Console.WriteLine();

return InputGroup(message);

}

}

static DateTime InputDateOfBirth(string message)

{

Console.WriteLine($"{message}");

if (DateTime.TryParse(Console.ReadLine(), out DateTime date))

{

return date;

}

else

{

Console.WriteLine("Некорректное значение. Введите заново.");

return InputDateOfBirth($"{message}");

}

}

static string InputPhoneNumber(string message)

{

Console.WriteLine(message);

string Phone = @"^\+7 \d{3} \d{3} \d{2} \d{2}$";

string input = Console.ReadLine();

if (Regex.IsMatch(input, Phone))

{

return input;

}

else

{

Console.WriteLine("Неккоретный ввод. Введите заново.");

return InputPhoneNumber(message);

}

}

enum Group

{

G1,

G2,

G3,

G4

}

struct Student

{

public Group Group;

public string LastName; //Фамилия

public string FirstName; //Имя

public string Surname; //Отчество

public DateTime DateOfBirht;

public string PhoneNumber;

}

static List<Student> list = new List<Student>();

static void Add\_To\_List()

{

try

{

Student st = new Student();

st.LastName = Input("Введите фамилию: ");

st.FirstName = Input("Введите имя: ");

st.Surname = Input("Отчество: ");

st.Group = (Group)InputGroup("Введите номер группы: G1, G2, G3, G4");

st.DateOfBirht = InputDateOfBirth("Введите дату рождения в формате: дд.мм.гггг (12.02.2002): ");

st.PhoneNumber = InputPhoneNumber("Введите номер телефона в правильном формате: +7 XXX XXX XX XX (+7 999 999 99 99):");

list.Add(st);

}

catch

{

Console.WriteLine("Ошибочные данные! Введите заново.");

Add\_To\_List();

}

}

static void Show\_List()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

for (int i = 0; i < list.Count; i++)

{

Console.WriteLine($"Номер студента в журнале: {i+1}");

Show(list[i]);

}

Console.WriteLine();

}

static void Delete()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

try

{

Console.Write("Введите номер студента в журнале, которого нужно отчислить: ");

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

list.RemoveAt(n - 1);

Console.WriteLine();

}

catch

{

Console.WriteLine("Ошибочные данные! Введите заново.");

Delete();

Console.WriteLine();

}

}

static void Show(Student st)

{

Console.WriteLine($"Фамилия: {st.LastName}");

Console.WriteLine($"Имя: {st.FirstName}");

Console.WriteLine($"Отчество: {st.Surname}");

Console.WriteLine($"Группа: {st.Group}");

Console.WriteLine($"Дата рождения: {st.DateOfBirht.ToString("dd.MM.yyyy")}");

Console.WriteLine($"Номер телефона: {st.PhoneNumber}");

Console.WriteLine("----------------------------");

}

static void Edit()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

string st = Input("Введите ФИО студента, которого хотите изменить:");

string[] FIO = st.Split(' ');

int m = 0;

bool ism = false;

int index = 0;

for (int i = 0; i < list.Count; i++)

{

if (FIO.Length == 3 &&

list[i].LastName.ToLower() == FIO[0].ToLower() &&

list[i].FirstName.ToLower() == FIO[1].ToLower() &&

list[i].Surname.ToLower() == FIO[2].ToLower())

{

index = i;

Console.WriteLine();

Show(list[i]);

do

{

Console.WriteLine("Введите номер поля, которое хотите изменить:\n1 - Фамилия\n2 - Имя\n3 - Отчество\n4 - Группа\n5 - Дата рождения\n6 - Номер телефона\n7 - Все поля\n8 - Ничего не хочу менять");

try

{

m = Convert.ToInt32(Console.ReadLine());

if (m < 1 || m > 8)

{

throw new FormatException();

}

ism = true;

}

catch (FormatException)

{

Console.WriteLine("Неверный ввод. Введите заново.");

}

} while (!ism);

}

}

Student student = list[index];

if (!ism) { Console.WriteLine("Студент с таким ФИО не был найден."); }

switch (m)

{

case 1:

student.LastName = Input("Введите новую фамилию: ");

list[index] = student;

break;

case 2:

student.FirstName = Input("Введите новое имя: ");

list[index] = student;

break;

case 3:

student.Surname = Input("Введите новое отчество: ");

list[index] = student;

break;

case 4:

student.Group = (Group)InputGroup("Введите новую группу: ");

list[index] = student;

break;

case 5:

student.DateOfBirht = InputDateOfBirth("Введите новую дату рождения в формате 12.02.2002: ");

list[index] = student;

break;

case 6:

student.PhoneNumber = InputPhoneNumber("Введите новый номер телефона в формате +7 999 999 99 99: ");

list[index] = student;

break;

case 7:

student.LastName = Input("Введите новую фамилию: ");

student.FirstName = Input("Введите новое имя: ");

student.Surname = Input("Введите новое отчество: ");

student.Group = InputGroup("Введите новую группу: ");

student.DateOfBirht = InputDateOfBirth("Введите новую дату рождения в формате 12.02.2002: ");

student.PhoneNumber = InputPhoneNumber("Введите новый номер телефона в формате +7 999 999 99 99: ");

list[index] = student;

break;

case 8:

break;

}

}

static void Search()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

Console.Write("Примечание! Для поиска студнтов по группе необходимо ввести группу в формате: GX (G1, G2, G3, G4)");

Console.Write("Введите значение поля, по которому хотите найти записи: ");

string inputField = "";

inputField = Console.ReadLine();

bool found = false;

for (int i = 0; i < list.Count; i++)

{

if (inputField == list[i].LastName)

{

Show(list[i]);

found = true;

}

else if (inputField == list[i].FirstName)

{

Show(list[i]);

found = true;

}

else if (inputField == list[i].Surname)

{

Show(list[i]);

found = true;

}

else if (Enum.TryParse<Group>(inputField, out Group group) && group == list[i].Group)

{

Show(list[i]);

found = true;

}

else if (DateTime.TryParse(inputField, out DateTime date))

{

if (date == list[i].DateOfBirht)

{

Show(list[i]);

found = true;

}

}

else if (inputField == list[i].PhoneNumber)

{

Show(list[i]);

found = true;

}

}

if (!found)

{

Console.WriteLine("По этому полю не было найдено ни одной записи.");

Console.WriteLine();

}

}

static void IsAdult()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

bool isAdult = false;

Group group = InputGroup("Для поиска студнтов по группе необходимо ввести группу в формате: GX (G1, G2, G3, G4)\nВведите группу список несовершеннолетних из которой хотите узнать: ");

Console.WriteLine($"Несовершеннолетние студенты группы {group}:");

for (int i = 0; i < list.Count; i++)

{

if (list[i].Group == group)

{

DateTime now = DateTime.Now;

DateTime dateBirth = list[i].DateOfBirht;

if (now.Subtract(dateBirth).TotalDays <= 18 \* 365)

{

Show(list[i]);

isAdult = true;

}

}

}

if (!isAdult) { Console.WriteLine($"В группе {group} все совершеннолетние."); }

}

static void LastnameWithLetter()

{

if (list.Count == 0) { Console.WriteLine("На данный момент не учиться ни одного студента."); return; }

char letter;

bool isSuccess = false;

do

{

Console.WriteLine("Введите букву, на которую начинается фамилия: ");

} while (!char.TryParse(Console.ReadLine(), out letter) && letter >= 'А' && letter <= 'я');

Console.WriteLine($"Студенты у которых фамилия начинается на '{letter}'");

for (int i = 0; i < list.Count; i++)

{

if (char.ToUpper(list[i].LastName[0]) == char.ToUpper(letter))

{

Show(list[i]);

isSuccess = true;

}

}

if (!isSuccess) { Console.WriteLine("Студентов с такой фамилией не было найдено("); }

}

static void Main()

{

// Студент 1

Student st1 = new Student();

st1.LastName = "Иванов";

st1.FirstName = "Иван";

st1.Surname = "Иванович";

st1.Group = Group.G1;

st1.DateOfBirht = new DateTime(2008, 1, 1);

st1.PhoneNumber = "+7 956 567 23 22";

// Студент 2

Student st2 = new Student();

st2.LastName = "Петров";

st2.FirstName = "Петр";

st2.Surname = "Петрович";

st2.Group = Group.G2;

st2.DateOfBirht = new DateTime(2001, 5, 15);

st2.PhoneNumber = "+7 934 234 55 67";

// Студент 3

Student st3 = new Student();

st3.LastName = "Сидоров";

st3.FirstName = "Сидор";

st3.Surname = "Сидорович";

st3.Group = Group.G3;

st3.DateOfBirht = new DateTime(1999, 10, 28);

st3.PhoneNumber = "+7 453 435 54 45";

list.Add(st1);

list.Add(st2);

list.Add(st3);

int m = 1;

do

{

bool ism = false;

do

{

try

{

Console.WriteLine(" Меню");

Console.WriteLine("1 - Показать список студентов");

Console.WriteLine("2 - Зачислить нового студента");

Console.WriteLine("3 - Отчислить студента");

Console.WriteLine("4 - Найти студента по ФИО и изменить его данные");

Console.WriteLine("5 - Поиск студентов по содержимому одного из полей");

Console.WriteLine("6 - Список несовершеннолетних");

Console.WriteLine("7 - Найти студентов по первой букве фамилии");

Console.WriteLine("8 - Выход");

m = Convert.ToInt32(Console.ReadLine());

if (m < 1 || m > 8)

{

throw new FormatException();

}

ism = true;

}

catch (FormatException)

{

Console.WriteLine();

Console.WriteLine("Неверный ввод. Введите заново.");

Console.WriteLine();

}

} while (!ism);

switch (m)

{

case 1:

Show\_List();

break;

case 2:

Add\_To\_List();

break;

case 3:

Delete();

break;

case 4:

Edit();

break;

case 5:

Search();

break;

case 6:

IsAdult();

break;

case 7:

LastnameWithLetter();

break;

case 8:

break;

}

} while (m < 8);

Console.ReadKey();

}

}

**Вывод программы:**

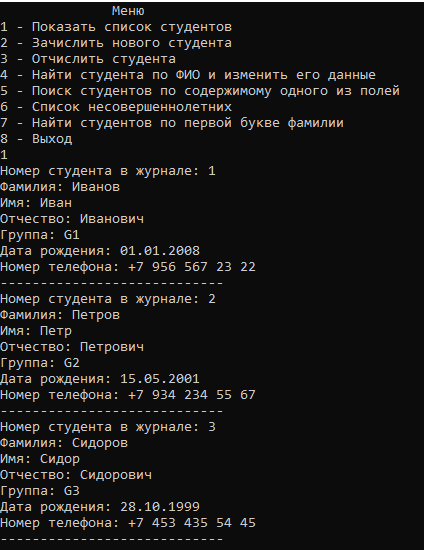
****

Рис.1 – Вывод

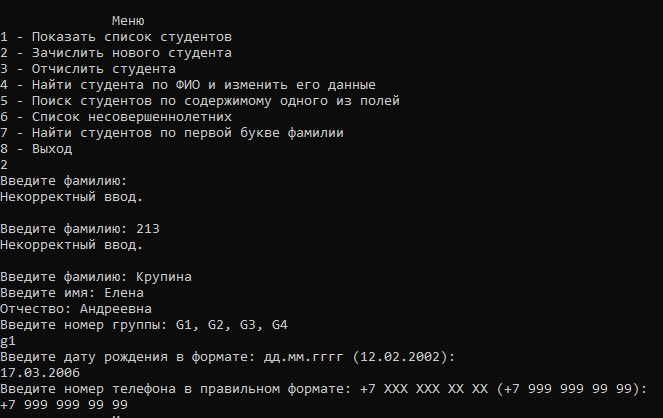


Рис.2 – Вывод

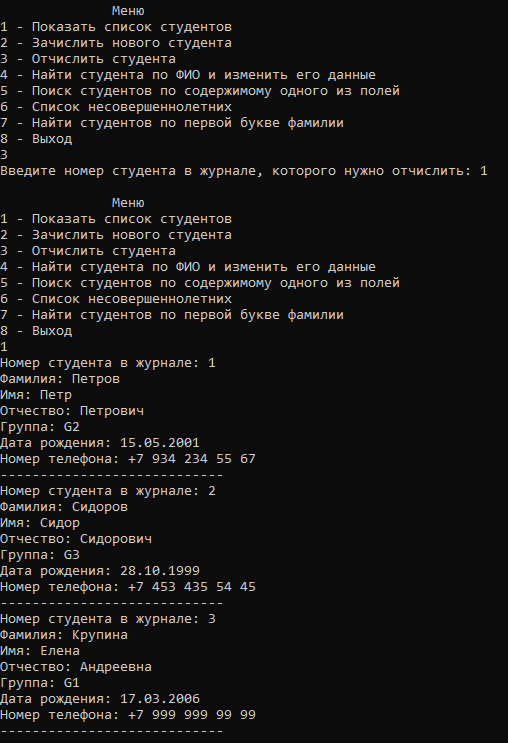


Рис.3 – Вывод

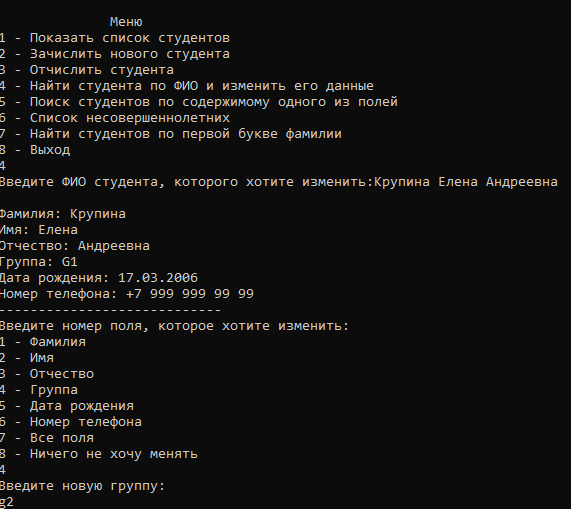


Рис.4 – Вывод

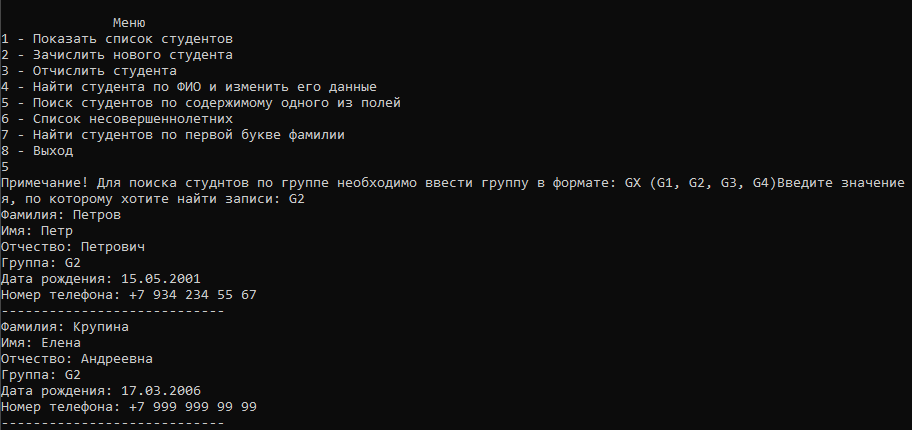


Рис.5 – Вывод