МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2024г.

Языки программирования

Методические указания (часть 2) по выполнению лабораторных работ для студентов, обучающихся по направлению 02.03.03 Математическое обеспечение и администрирование информационных систем

Курск 2024

2

УДК 681.3(075)

Составитель Л. А. Лисицин

Рецензент
Кандидат технических наук, доцент *Халин Ю.А.*

Языки-программирования [Текст]: методические указания (часть 2) по выполнению лабораторных работ по направлениям 02.03.03 Математическое обеспечение и администрирование информационных систем / Юго-Зап. гос. ун-т; сост.: Л. А. Лисицин. Курск, 2024. 62 с.: ил. 8. табл. 3. Библиогр. с. 62.

Содержат основные теоретические положения и приемы разработки программ в интегрированной среде MS Visual Studio C#, пример решения типовой задачи, индивидуальные задания и контрольные вопросы к защите лабораторной работы. Отражен порядок выполнения лабораторных работ и правила оформления отчетов.

Методические указания предназначены для студентов, обучающихся по направлению 02.03.03 Математическое обеспечение и администрирование информационных систем.

Текст печатается в авторской редакции

Подписано в печать 9.04 Формат 60x84 1/16.
Усл.печ. л. 3,3 Уч.-изд. л. 3,1. Тираж 100 экз. Заказ 586. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Оглавление

Лабораторная работа №8: «Строки на языке C#»

Цель работы – изучение символов и строк. Строки, как массивы символов. Типы char и string.

Основные понятия

Символьный тип относится к встроенным типам данных C# и соответствует стандартному классу Char библиотеки .NET из пространства имен System. В этом классе определены а также преобразовать символ в верхний или нижний регистр и в число. Основные методы приведены в таблице.

Таблица 1. Основные методы класса System.Char

Метод	Описание
GetNumericValue	Возвращает числовое значение символа, если он является цифрой, и -1 в противном случае
GetUnicodeCategory	Возвращает категорию Unicode-символа1
IsControl	Возвращает true, если символ является управляющим
IsDigit	Возвращает true, если символ является десятичной цифрой
IsLetter	Возвращает true, если символ является буквой
IsLetterOrDigit.	Возвращает true, если символ является буквой или цифрой
IsLower	Возвращает true, если символ задан в нижнем регистре
IsNumber	Возвращает true, если символ является числом (десятичным или шестнадцатеричным)
IsPunctuation	Возвращает true, если символ является знаком препинания
IsSeparator	Возвращает true, если символ является разделителем
IsUpper	Возвращает true, если символ записан в верхнем регистре
IsWhiteSpace	Возвращает true, если символ является пробельным (пробел, перевод строки и возврат каретки)
Parse	Преобразует строку в символ (строка должна состоять из одного символа)
ToLower	Преобразует символ в нижний регистр
ToUpper	Преобразует символ в верхний регистр

MaxValue, MinValue	Возвращают символы с максимальным и минимальным кодами (эти символы не имеют видимого представления)

В листинге 6.6 продемонстрировано использование этих методов.

Листинг 1. Использование методов класса System.Char:

```
using System;
namespace ConsoleApplicationl { class Classl
{ static void MainO {
try     '
{
char b = 'B\ c = '\x63', d = '\u0032';                    (1)
Console.WriteLineC "{0} {1} {2}", b, c, d );
Console.WriteLineC "{0} {1} {2}",
char.ToLower(b), char.Tollpper(c), char.GetNumericValue(d) );
char a;
do
{
Console.WriteC "Введите символ: " ); a = char.Parse( Console.ReadLineO );
Console.WriteLineC "Введен символ {0}, его код - {1}", a, (int)a ):
if (char.IsLetter(a)) Console.WriteLineC'ByKBa"); if (char.Isllpper(a))
Console.WriteLineC'BepxHnn per."):
if (char.IsLower(a))        Console.WriteLineC"Нижний per.");
if (char.IsControl(a)) Console.WriteLine("Управляющий"); if
(char.IsNumber(a)) Console.WriteLineOHn^o"); if (char.IsPunctuation(a))
Console.WriteLine("Разделитель");
} while (a != 'q');
catch
{
Console.WriteLine( "Возникло исключение" );
return;
```

В операторе 1 описаны три символьных переменных. Они инициализируются символьными литералами в различных формах представления. Далее выполня¬ются вывод и преобразование символов.

В цикле 2 анализируется вводимый с клавиатуры символ. Можно вводить и управ¬ляющие символы, используя сочетание клавиши Ctrl с латинскими буквами. При вводе использован метод Parse, преобразующий строку, которая должна содер¬жать единственный символ, в символ типа char. Поскольку вводится строка, ввод каждого символа следует завершать нажатием клавиши Enter. Цикл выполняет¬ся, пока пользователь не введет символ q.

Вывод символа сопровождается его кодом в десятичном виде1. Для вывода кода используется явное преобразование к целому типу. Явное преобразование из символов в строки и обратно в C# не существует, неявным же образом любой объект, в том числе и символ, может быть преобразован в строку2, например:

```
string s = 'к' + 'о' + 'т';     // результат - строка "кот"
```

При вводе и преобразовании могут возникать исключительные ситуации, например, если пользователь введет пустую строку. Для «мягкого» завершения программы предусмотрена обработка исключений.

Массив символов, как и массив любого иного типа, построен па основе базового класса Array, некоторые свойства и методы которого были перечислены в? табл. 6.1. Применение этих методов позволяет эффективно решать некоторые задачи. Простой пример приведен в листинге 6.7

Листинг 2. Работа с массивом символов:

```
using System;
namespace ConsoleApplicationl { class Classl
{ static void MainO {
,char[] a = { 'm\ 'a', 's', 's', 'i\ 'v' };       // 1
char[] b = "a роза упала на лапу азора" .ToCharArrayO; // 2
PrintArray( "Исходный массив a:", a );
int pos = Array.IndexOf( a. 'm' ); a[pos] = 'M';
PrintArray( "Измененный массив a:", a );
PrintArray( "Исходный массив b:". b );
Array.Reverse( b );
PrintArray( "Измененный массив b:\ b );
}
public static void PrintArray( string header. Array a )
{
Console.WriteLine( header );
foreach ( object x in a ) Console.Write( x );
Console.WriteLine( "\n" );
}
}
}
```

Результат работы программы:

```
Исходный массив a: massiv
Измененный массив a:
Massiv        '
Исходный массив b: a роза упала на лапу азора
```

Измененный массив b: ароза упал ан алапу азор а

Символьный массив можно инициализировать, либо непосредственно задавая его элементы (оператор 1), либо применяя метод ToCharArray класса string, который разбивает исходную строку на отдельные символы (оператор 2).

Строки типа string

Тип string, предназначенный для работы со строками символов в кодировке Unicode, является встроенным типом C#. Ему соответствует базовый класс System.String библиотеки .NET.

Создать строку можно несколькими способами:

```
string s;       // инициализация отложена
string t = "qqq"':    11 инициализация строковым литералом
string u = new. stringC \ 20):    // конструктор создает строку из 20 пробелов
char[] a={ 'O', 'O', '0' }:   // массив для инициализации строки
string v = new string( a );          // создание из массива символов
```

Для строк определены следующие операции:

- □ присваивание (=);
- □ проверка на равенство (==);
- □ проверка на неравенство (! =);
- □ обращение по индексу ([]);
- □ сцепление (конкатенация) строк (+).

Несмотря на то что строки являются ссылочным типом данных, на равенство и неравенство проверяются не ссылки, а значения строк. Строки равны, если име¬ют одинаковое количество символов и совпадают посимвольно.

Обращаться к отдельному элементу строки по индексу можно только для полу¬чения значения, но не для его изменения. Это связано с тем, что строки типа string относятся к .так называемым неизменяемым типам данныхК Методы, из¬меняющие содержимое строки, на самом деле создают новую копию строки. Не¬используемые «старые» копии автоматически удаляются сборщиком мусора.

В классе System.String предусмотрено множество методов, полей и свойств, поз¬воляющих выполнять со строками практически любые действия. Основные эле¬менты класса приведены в табл. 6.3.

Таблица 2. Основные элементы класса System.String

Название	Вид	Описание
Compare	Статический метод	Сравнение двух строк в лексикографическом (алфавитном) порядке. Разные реализации метода позволяют сравнивать строки и подстроки с учетом и без учета регистра и особенностей национального представления дат и т. д.

Название	Вид	Описание
CompareOrdinal	Статический метод	Сравнение двух строк по кодам символов. Разные реализации метода позволяют сравнивать строки и подстроки
CompareTo	Метод	Сравнение текущего экземпляра строки с другой строкой,
Concat	Статический метод	Конкатенация строк. Метод допускает сцепление произвольного числа строк
Copy	Статический метод	Создание копии строки
Empty	Статическое поле	Пустая строка (только для чтения)
Format	Статический метод	Форматирование в соответствии с заданными спецификаторами формата (см. далее)
IndexOf, IndexOfAny, LastlndexOf, LastlndexOfAny	Методы	Определение индексов первого и последнего вхождения заданной подстроки или любого символа из заданного набора
Insert	Метод	Вставка подстроки в заданную позицию
Intern,	Статические методы	Возвращает ссылку на строку, если такая
Islnterned		уже существует. Если строки нет, Intern добавляет строку во внутренний пул, Islnterned возвращает null
Join	Статический метод	Возвращает ссылку на строку, если такая уже существует. Если строки нет, Intern добавляет строку во внутренний пул, Islnterned возвращает null
Length	Свойство	Длина строки (количество символов)
PadLeft,PadRight	Методы	Выравнивание строки по левому или правому краю путем вставки нужного числа пробелов в начале или в конце строки
Remove	Метод	Удаление подстроки из заданной позиции
Replace	Метод	Замена всех вхождений заданной подстроки или символа новыми подстрокой или символом
Split	Метод	Разделяет строку на элементы, используя заданные разделители. Результаты помещаются в массив строк
Substring	Метод	Выделение подстроки, начиная с заданной позиции
ToCharArray	Метод	Преобразование строки в массив символов
ToLower, Tollpper	Методы	Преобразование символов строки к нижнему или верхнему регистру

Пример применения методов приведен в листинге 3.

Листинг 3. Работа со строками типа string:

```
using System;
namespace ConsoleApplicationl,
```

```
{ class Classl
{ static void MainO
{ •
string s - "прекрасная королева Изольда";
Console.WriteLineC s );
string sub = s.SubstringC 3 ).Remove( 12, 2 );       II 1
Console.WriteLineC sub ); •
string[] mas = s.SplitO ');•         // 2
string joined = string.Join( "! ", mas );
Console.WriteLine( joined );
Console.WriteLineC "Введите строку" );
string x = Console.ReadLineO; // 3
Console.WriteLineC "Вы ввели строку " + x );
double a = 12.234;
int b = 29;
Console.WriteLineC " a = {0,6:C} b = {1,2:X}\ a, b );
Console.WriteLineC " a = {0,6:0.##} b = {1,5:0.# ' руб
a, b );
Console'.WriteLineC" a = {0:F3} b = {1:D3}", a, b,);
}
}
}
```

Результат работы программы:

```
прекрасная королева Изольда
красная корова Изольда
прекрасная! королева! Изольда
Введите строку
не хочу!
Вы ввели строку не хочу!
a = 12,23р. b = 1D
a = 12,23 b = 29 руб.
```

В операторе 1 выполняются два последовательных вызова методов: метод Substring возвращает подстроку строки s, которая содержит символы исходной строки, начиная с третьего. Для этой подстроки вызывается метод Remove, удаляющий из нее два символа, начиная с 12-го. Результат работы метода присваивается переменной sub.

При работе со строками необходимо учитывать, что в C# строка типа string является неизменяемым типом данных, то есть любая операция изменения

строки на самом деле возвращает ее копию. Для изменения строк используется тип StringBuilder. Прежде чем описывать в программе какое-либо действие со строками, полезно посмотреть, нет ли в списке элементов используемого класса подходящих методов и свойств.

Для эффективного поиска и преобразования текста в соответствии с заданными шаблонами используются так называемые регулярные выражения.

Задание: с помощью текстового редактора создать файл, содержащий текст, длина которого не превышает 1000 символов (длина строки текста не должна превышать 70 символов). Имя файла должно иметь расширение DAT. Слова в тексте разделены одним или несколькими пробелами. В конце каждого предложения текста ставится точка. Написать с применением интегрированной среды разработки консольное приложение, которое в соответствии с номером варианта задания выполняет работу с созданным текстовым файлом.

Варианты заданий:

1. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество предложений в тексте.

2. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте.

3. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, оканчивающихся на гласную букву.

4. Написать программу, которая:
- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, располагая слова текста в обратном порядке и удаляя лишние пробелы.

5. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, у которых первый и последний символы совпадают.

6. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, начинающихся на гласную букву.

7. Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет количество символов в самом длинном слове.

8. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество символов в самом коротком слове.

9. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет в каждом предложении текста количество символов, отличных от букв и пробела.

10. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество предложений текста и количество слов в каждом предложении.

11. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество букв 'a' в последнем слове текста.

12. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет самую длинную последовательность цифр в тексте (считать, что любое количество пробелов между двумя цифрами не прерывает последовательности цифр).

13. Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет порядковый номер заданного слова в каждом предложении текста (заданное слово вводится пользователем).

14. Написать программу, которая:
- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, выкидывая из него заданное слово (заданное слово вводится пользователем) и удаляя лишние пробелы.

15. Написать программу, которая:
- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, меняя в нем местами заданные слова (заданные слова вводятся пользователем) и удаляя лишние пробелы.

16. Написать программу, которая:
- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, заключая заданное слово (заданное слово вводится пользователем) в кавычки.

17. Написать программу, которая:

- выводит текст файла на экран монитора;

- выводит текст на экран монитора еще раз, вставляя в каждое предложение в качестве последнего заданное слово (заданное слово вводится пользователем).

18. Написать программу, которая:

- выводит текст файла на экран монитора;

- выводит текст на экран монитора еще раз, убирая лишние пробелы между словами и начиная каждое предложение с новой строки.

19. Написать программу, которая:

- выводит текст файла на экран монитора;

- выводит текст на экран монитора еще раз, заменяя в заданном слове (заданное слово вводится пользователем) строчные буквы прописными.

20. Написать программу, которая:

- выводит текст файла на экран монитора;

- определяет наибольшее количество подряд идущих пробелов в тексте.

Лабораторная работа №9: «Графический режим работы на языке C#»

Цель работы — изучение графического режима работы. Графические примитивы.

Методические указания

Программа должна иметь одно окно (Form1). Для организации формирования изображения на канве окна приложения (Form1->Canvas) необходимо на форме окна разместить объект TTimer, создав обработчик события OnTimer и установив интервал срабатывания события Timer1->Interval = 1. Обработчик события OnTimer должен выполнять вывод текста и фигур в окно приложения. Для поворота текста или графической фигуры относительно границ окна приложения воспользуйтесь функциями SetGraphicsMode() и SetWorldTransform(). Чтобы вывести на канву окна приложения текст, необходимо использовать функцию TextOutA(). Для задания цвета используйте функцию RGB(r, g, b), где r, g и b – компоненты, соответствующие цветовой модели RGB: канал красного цвета (r), канал зеленого цвета (g) и канал синего цвета (b). При построении линий и контуров графических фигур используется объект «перо» (Form1->Canvas->Pen), при «заливке» цветом и узором замкнутых контуров фигур – объект «кисть» (Form1->Canvas->Brush).

Пример текста программы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>
 #include "Unit1.h"
 #pragma package(smart_init)
 #pragma resource "*.dfm"
 TForm1 *Form1;
```

```
 int r = 0, g = 0, b = 0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
} //---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
Form1->Canvas->Pen->Style = psClear;
this->Canvas->Brush->Color = RGB(50, 200, 100);
Canvas->Ellipse(100, 100, 500, 500);
 Canvas->Brush->Color = clYellow;
Canvas->Brush->Style = bsSolid;
Canvas->Chord(100, 100, 500, 500, 500, 100, 100,100);
 Timer1Timer(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
SetGraphicsMode(Canvas->Handle, GM_ADVANCED);
XFORM xform; xform.eM11 = cos(0.2);
xform.eM12 = -sin(0.2); xform.eM21 = sin(0.2);
xform.eM22 = cos(0.2);
xform.eDx = (float) 100; xform.eDy = (float) 150;
 SetWorldTransform(Canvas->Handle, &xform);
 r = r + 10;
```

```
g = r + g;
b = (r * 40) + g + b;
 if (r >= 255) r = 0;
if (g >= 255) g = 0;
if (b >= 255) b = 0;
 Canvas->Brush->Color = RGB(255, 255, 255);
Canvas->Brush->Style = bsSolid;
 Canvas->Font->Color = RGB(r, g, b);
Canvas->Font->Name = "Arial";
Canvas->Font->Style = Canvas->Font->Style << fsItalic << fsBold;
Canvas->Font->Size = 27;
Canvas->TextOutA(100, 200, "Student");
 Form1->Canvas->Pen->Style = psDash;
Form1->Canvas->Pen->Color = clBlack;
Form1->Canvas->Pen->Width = 1;
Canvas->MoveTo(100, 100);
Canvas->LineTo(200, 200); Canvas->LineTo(100, 200);
} //
```

Задание: Написать с применением интегрированной среды разработки приложение с графическим интерфейсом пользователя ОС Microsoft Windows, которое в соответствии с номером варианта задания выполняет вывод текста и фигур в окно.

Варианты заданий:

1. В окно бледно-зеленого цвета выведите три красных круга диаметром 200 пикселей, расположенные пирамидой (один круг в центре над двумя другими). Выведите поверх изображения свою

фамилию шрифтом Times, размером 80 пикселей, перечеркнутым, синего цвета. Надпись расположите горизонтально.

2. В окно голубого цвета выведите три концентрических квадрата с размерами сторон 300, 200 и 100 пикселей. Внешний квадрат нарисуйте толстым зеленым пером, средний – фиолетовым, внутренний – красным. Квадраты должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Arial, размером 80 пикселей, жирным, желтого цвета. Надпись расположите под углом 15° к горизонтали.

3. В окно желтого цвета выведите три соприкасающихся круга диаметром 200 пикселей каждый, расположенные горизонтально и залитые красным, зеленым и синим цветами. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, курсивом, черного цвета. Надпись расположите под углом 45° (снизу вверх).

4. В окно красного цвета выведите залитый синим цветом квадрат, ориентированный под углом 45° к осям координат. Такой наклонный квадрат рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, жирным, желтого цвета. Надпись расположите вертикально (снизу вверх).

5. В окно желтого цвета выведите три концентрические окружности с диаметрами 400, 300 и 200 пикселей. Внешнюю окружность нарисуйте толстым (6-8 пикселей) зеленым пером, среднюю – синим, а внутреннюю – коричневым. Образованные круги должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, курсивом, синего цвета. Надпись расположите вертикально (сверху вниз).

6. В окно серого цвета выведите круг розового цвета диаметром 300 пикселей, у которого правая средняя четверть (сектор размером 90°) покрашена в синий цвет. Выведите поверх изображения свою

фамилию шрифтом Courier размером 80 пикселей, жирным, красного цвета. Надпись расположите под углом 45° (сверху вниз).

7. В окно голубого цвета выведите три концентрических круга с диаметрами 300, 200 и 100 пикселей. Закрасьте внешний круг зеленым цветом, средний – желтым, а внутренний – белым. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, курсивом, темно-синего цвета. Надпись расположите горизонтально.

8. В окно желтого цвета выведите круг диаметром 400 пикселей, верхняя половина которого закрашена синим, а нижняя - красным цветом. Части окружности рисуются с помощью функций Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию вертикально снизу вверх шрифтом Courier размером 80 пикселей, жирным, перечеркнутым, белого цвета.

9. В окно черного цвета выведите три концентрических квадрата с размерами сторон 400, 300 и 200 пикселей. Внешний квадрат закрасьте желтым цветом, средний – синим, а внутренний – красным. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, подчеркнутым, светлосерого цвета. Надпись расположите горизонтально, «вверх ногами».

10. В окно светло-зеленого цвета выведите круг диаметром 400 пикселей, залитый желтым цветом, верхняя четверть которого (до соответствующей хорды) закрашена темно-зеленым цветом. Фигуры, ограниченные частью окружности и хордой, рисуются с помощью функции Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, жирным, курсивом, синего цвета. Надпись расположите под углом 30° к горизонтали.

11. В окно светло-коричневого цвета выведите толстую (6-8 пикселей) фиолетовую окружность диаметром 400 пикселей, левая половина которой заштрихована горизонтально, а правая - вертикально тем же фиолетовым цветом. Части окружности рисуются с помощью функций Pie() или Chord() класса TCanvas. Кисть для штриховки замкнутой

фигуры создается конструктором TBrush с указанием стиля, значения которого можно найти в описании функции CreateHatchBrush(). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, курсивом, синего цвета. Надпись расположите вертикально сверху вниз.

12. В окно светло-голубого цвета выведите желтый круг диаметром 400 пикселей, с «выеденной» с правой стороны частью (как у надкусанного яблока). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, подчеркнутым, жирным, курсивом, красного цвета. Надпись расположите под углом 75° (снизу вверх).

13. В окно коричневого цвета выведите три соприкасающихся круга диаметром 150 пикселей каждый, расположенные вертикально и залитые серым, красным и белым цветами. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, подчеркнутым и перечеркнутым, черного цвета. Надпись расположите вертикально снизу вверх.

14. В окно серого цвета выведите три вложенных друг в друга квадрата с размерами сторон 300, 200 и 100 пикселей, соприкасающиеся левыми верхними углами. Внешний квадрат закрасьте фиолетовым цветом, средний - темно-синим, внутренний – светло-синим. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, жирным, желтого цвета. Надпись расположите под углом 45° (снизу вверх).

15. В окно синего цвета выведите круг диаметром 400 пикселей желтого цвета, у которого верхняя левая четверть (сектор размером 90°) прозрачна. Сектор рисуется с помощью функции Pie() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, перечеркнутым, жирным, красного цвета. Надпись расположите под углом 75° (снизу вверх).

16. В окно желтого цвета выведите два эллипса синего цвета с общим центром, расположенные один горизонтально, а другой вертикально. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, узким (шириной 10 пикселей), жирным, красного цвета. Надпись расположите горизонтально.

17. В окно бледно-зеленого цвета выведите круг диаметром 400 пикселей, левая половина которого закрашена темно-красным, а правая - ярко-красным цветом. Полукруги рисуются с помощью функций Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, подчеркнутым, жирным, светло-серого цвета. Надпись расположите под углом 15° сверху вниз.

18. В окно светло-бирюзового цвета выведите круг диаметром 400 пикселей, разделенный пополам диагональю, проходящей вправо и вверх. Левую половину круга закрасьте красным цветом, правую – синим. Полукруги рисуются с помощью функций Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, широким (шириной 80 пикселей), курсивом, черного цвета. Надпись расположите горизонтально.

19. В окно светло-серого цвета выведите прямоугольный треугольник синего цвета с длиной катетов 400 пикселей. Треугольник рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, курсивом, белого цвета. Надпись расположите под углом 75°.

20. В окно бледно-розового цвета выведите три концентрических эллипса с размерами 500x300, 400x200 и 200x100 пикселей. Внешний эллипс нарисуйте толстым (6-8 пикселей) черным пером, средний - синим, а внутренний - желтым. Эллипсы должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, перечеркнутым, узким (шириной 10 пикселей), красного цвета. Надпись расположите горизонтально.

21. В окно голубого цвета выведите изображение Солнца в виде желтого круга, эллиптической орбиты Земли вокруг него и самой Земли в виде коричневого кружка в верхней точке этой орбиты. Выведите поверх изображения горизонтально свою фамилию шрифтом Courier размером 20 пикселей, подчеркнутым, белого цвета. Надпись расположите горизонтально.

22. В окно синего цвета выведите большой желтый треугольник. Треугольник рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, узким (шириной 20 пикселей), черного цвета. Надпись расположите под углом 45° сверху вниз.

23. В окно синего цвета выведите изображение красного полукруглого солнца (как бы заходящего за горизонт) с расходящимися от него несколькими лучами в виде толстых красных линий. Линии рисуются функцией LineTo(), начальная точка перемещается функцией MoveTo(). Половины круга рисуются функциями Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, курсивом, светло-серого цвета. Надпись расположите горизонтально вверх ногами.

24. В окно зеленого цвета выведите друг над другом два полукруга, соприкасающихся своими округлыми частями. Верхний полукруг залейте синим цветом, нижний - красным. Половины круга рисуются функциями Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, подчеркнутым и перечеркнутым, белого цвета. Надпись расположите горизонтально.

25. В окно светло-голубого цвета выведите желтое изображение солнца с расходящимися от него несколькими толстыми желтыми лучами. Линии рисуются функцией LineTo(), начальная точка перемещается функцией MoveTo() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, перечеркнутым, жирным, курсивом, красного цвета. Надпись расположите под углом 30° снизу вверх.

26. В окно голубого цвета выведите удлиненный ромб, расположенный горизонтально и залитый темно-красным цветом. Ромб рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 150 пикселей, белого цвета. Надпись расположите вертикально (сверху вниз).

27. В окно бледно-зеленого цвета выведите фигуру наподобие песочных часов – два узких сектора, соединенные вершинами и нарисованные толстым белым пером. Секторы должны казаться

прозрачными. Секторы рисуются вызовом функции Pie() класса TCanvas. «Насыпьте» на дно песочных часов немного желтого песка (воспользуйтесь для этого функцией Chord() класса TCanvas). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, узким (шириной 15 пикселей), красного цвета. Надпись расположите горизонтально вверх ногами.

28. В окно коричневого цвета выведите два соприкасающихся круга диаметром 200 пикселей каждый, расположенные по диагонали и залитые зеленым и синим цветами. Выведите поверх изображения свою фамилию шрифтом Courier небольшого размера (30 пикселей), но большой ширины (80 пикселей), жирным, красного цвета. Надпись расположите под углом 30° снизу вверх.

29. В окно светло-голубого цвета выведите три круга, расположенные на одной горизонтальной прямой так, что каждый следующий круг сдвинут относительно предыдущего на величину своего радиуса. Первый круг покрасьте в красный цвет, второй - в белый, а третий - в зеленый. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, черного цвета. Надпись расположите под углом 5° сверху вниз.

30. В окно желтого цвета выведите неправильный выпуклый пятиугольник темно-синего цвета. Такая фигура рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения

свою фамилию шрифтом Arial размером 10 пикселей, красного цвета. Надпись расположите горизонтально «вверх ногами».

Лабораторная работа №10: «Структуры и объединения на языке C#»

Цель работы – Программирование в системе C# с использованием структур. Основные понятия. Варианты с использования структур в программировании.

Основные понятия

Структуры

С целью повышения эффективности программы, в C# предусмотрена структура, которая подобна классу, но относится к типу значения, а не к ссылочному типу данных. Ведь каждый доступ к объектам (даже самым мелким) по ссылке связан с дополнительными издержками на расход вычислительных ресурсов и оперативной памяти

Структуры объявляются с помощью ключевого слова struct и с точки зрения синтаксиса подобны классам. Ниже приведена общая форма объявления структуры:

struct имя : интерфейсы {

// объявления членов

}

где имя обозначает конкретное имя структуры.

В декларации структуры struct – служебное слово. Далее идут имя и интерфейсы структуры. За этим уже следует тело структуры, заключающееся в фигурные скобки, которое представляет последовательность объявлений членов структуры. Структура может иметь модификаторы, такие как: new, public, protected, internal, private.

Структура может имеет следующий формат:

Модификатор struct Имя Интерфейс

Тело

Стоит сказать сразу, что *структуры не участвуют в наследовании*, однако, они могут служить реализацией интерфейсов.

Членами структуры могут быть: константы, поля, методы, свойства, индексаторы, события, операции, конструкторы экземпляров, статические конструкторы и вложенные типы. В отличии от классов членом структуры не может быть *финализатор и объявление вложенной структуры.*

Прежде чем привести пример, отметим некоторые отличия структур от классов. Во-первых, в объявлении поля структуры *нельзя использовать инициализатор*. Т.е. нельзя писать так: double x =5;

Во-вторых, в каждое объявление структуры автоматически встраивается конструктор умолчания без параметров. Назначение данного конструктора – инициализировать все члены структуры значениями, которые соответствуют их типам по умолчанию. Например, для арифметических типов конструктор ставит нули, а для ссылок null. Поэтому в структуре нельзя явно объявить конструктор умолчания без параметров.

Например, определим структуру, которая описывает некоторые характеристики человека:

```
struct User
{
   public string name; // переменная
   public int age; // переменная
   public void DisplayInfo() // метод
   {
      Console.WriteLine($"Name: {name}  Age: {age}");
   }
}
```

Как и классы, структуры могут хранить состояние в виде переменных и определять поведение в виде методов. Так, в данном случае определены две переменные - name и age для хранения соответственно имени и возраста человека и метод DisplayInfo для вывода информации о человеке.

Пример использования структуры в программе:

```
using System;
namespace Program
```

```
{
    struct User // Блок структуры
    {
        public string name;
        public int age;
        public void DisplayInfo()
        {
            Console.WriteLine($"Name: {name}  Age: {age}");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            User ivan; // Создание объекта
            ivan.name = "Ivan"; // Присвоение объекту ivan имени
            ivan.age = 34; // Присвоение объекту ivan возраста
            ivan.DisplayInfo(); // Вывод информации об объекте
            Console.ReadKey();
        }
    }
}
```

В данном случае создается объект ivan. У него устанавливаются значения глобальных переменных, и затем выводится информация о нем.

Конструкторы структуры

Как и класс, структура может определять констукторы. Но в отличие от класса нам не обязательно вызывать конструктор для создания объекта структуры:

```
User ivan;
```

Однако, обязательно надо проинициализировать все поля (глобальные переменные) структуры перед получением их значений или перед вызовом методов структуры. То есть, например, в

следующем случае мы получим ошибку, так как обращение к полям и методам происходит до присвоения им начальных значений:

```
User ivan;
int x = ivan.age;    // Ошибка
ivan.DisplayInfo();  // Ошибка
```

Также, как говорилось ранее, мы можем использовать для создания структуры конструктор по умолчанию, при вызове которого полям структуры будет присвоено значение по умолчанию:

```
User ivan = new User();
ivan.DisplayInfo();
```

При этом мы получим следующие значения: Name: Age: 0

Мы можем определять свои конструкторы. Изменим структуру User:

```
struct User
{
   public string name;
   public int age;
   public User(string name, int age) // Создание конструктора
   {
    this.name = name;
/* Ключевое слово this является ссылкой на текущий объект как на самого себя */
        this.age = age;
   }
   public void DisplayInfo()
   {
      Console.WriteLine($"Name: {name}  Age: {age}");
   }
}

class Program
{
   static void Main(string[] args)
   {
```

```
            User ivan = new User("Ivan ", 21);
            ivan.DisplayInfo(); // Name = Ivan        Age = 21

            User nikolai = new User();
            nikolai.DisplayInfo(); // Name =       Age = 0

            Console.ReadKey();
        }
    }
```

Важно учитывать, что если мы определяем конструктор в структуре, то он должен инициализировать все поля структуры, как в данном случае устанавливаются значения для переменных name и age.

Также, как и для класса, можно использовать инициализатор для создания структуры:

```
User person = new User { name = "Sam", age = 31 };
```

Пример программирования с использованием структур

Задание.

Имеются данные о сотрудниках фирмы (фамилия, зарплата, пол). Найти фамилию мужчины, имеющего самую высокую зарплату. Если таких сотрудником несколько, то вывести всех, упорядочив список фамилий по убыванию.

Вид формы

Текст программы.

```
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 WindowsFormsApp4
{
   public partial class Form1 : Form
   {
      private bool flag = false;
      public bool Flag { get => flag; set => flag = value; }
      private int n;
      private uint p;
      public uint P { get => p; set => p = value; }
      private string f;
      public string F { get => f; set => f = value; }
      private string g;
      public string G { get => g; set => g = value; }
      public int N { get => n; set => n = value; }

      public Form1()
      {
         InitializeComponent();
      }
      struct firm
      {
         string surname;
         uint pay;
         string gender;
          public string Surname { get { return surname; } set { surname
= value; } }
          public uint Pay { get { return pay; } set { pay = value; } }
          public string Gender { get { return gender; } set { gender =
value; } }
         }

      private void button1_Click(object sender, EventArgs e)
      {
```

```
            firm[] person;
            person = new firm[4];
            person[0].Surname = "Volkov";
            person[0].Pay = 29400;
            person[0].Gender = "male";
            person[1].Surname = "Volkova";
            person[1].Pay = 59300;
            person[1].Gender = "female";
            person[2].Surname = "Petrov";
            person[2].Pay = 23600;
            person[2].Gender = "male";
            person[3].Surname = "Ivanov";
            person[3].Pay = 44100;
            person[3].Gender = "male";
            uint max = person[0].Pay;
            for (int i = 0; i < 3; i++)
            {
               if ((person[i].Gender == "male") && (person[i + 1].Gender
== "male") && (max < person[i + 1].Pay))
               {
                  N = i + 1;
                  max = person[N].Pay;
               }
            }
            for (int i = 0; i < 4; i++)
            { if ((person[i].Gender == "male") && (max == person[i].Pay)
&& (i != N))
               {
                  Flag = true;
                  break;
               }

            }
            if (Flag == false)
```

```
            {
                dataGridView1.Rows.Add();
                dataGridView1[0, 1].Value = person[N].Surname;
                dataGridView1[1, 1].Value = person[N].Pay;
                dataGridView1[2, 1].Value = person[N].Gender;
            }
            else
            {
                for (int k = 3; k > 0; k--)
                    for (int i = 0; i < k; i++)
                        if (person[i].Pay > person[i + 1].Pay)
                        {
                            P = person[i].Pay;
                            person[i].Pay = person[i + 1].Pay;
                            person[i + 1].Pay = P;
                            F = person[i].Surname;
                            person[i].Surname = person[i + 1].Surname;
                            person[i + 1].Surname = F;
                            G = person[i].Gender;
                            person[i].Gender = person[i + 1].Gender;
                            person[i + 1].Gender = G;
                        }
                for (int i = 0; i < 4; i++)
                {
                    dataGridView1.Rows.Add();
                    dataGridView1[0, i].Value = person[i].Surname;
                    dataGridView1[1, i].Value = person[i].Pay;
                    dataGridView1[2, i].Value = person[i].Gender;
                }
            }
        }
    }
```

Работа программы(Рисунки 1-2).

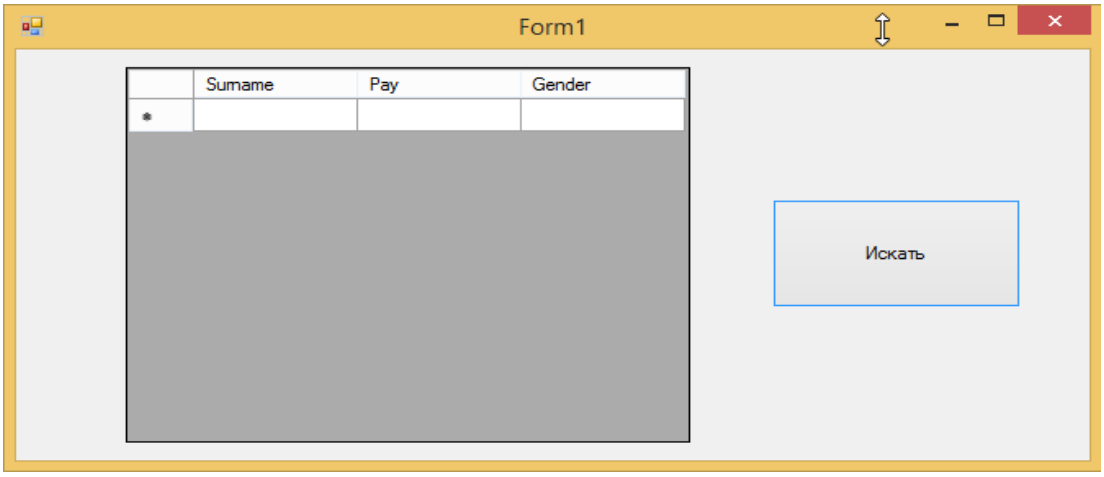

Рисунок 1 – Создание структуры

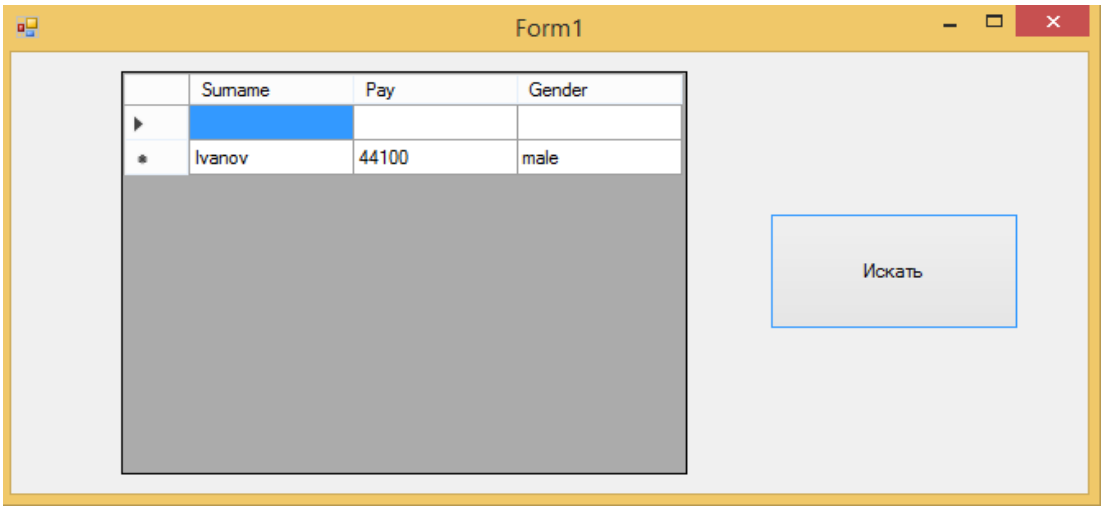

Рисунок 2 – Заполнение структуры

И работа при условии, если несколько сотрудников мужчины имеющие самую большую зарплату (Рисунок 13.)

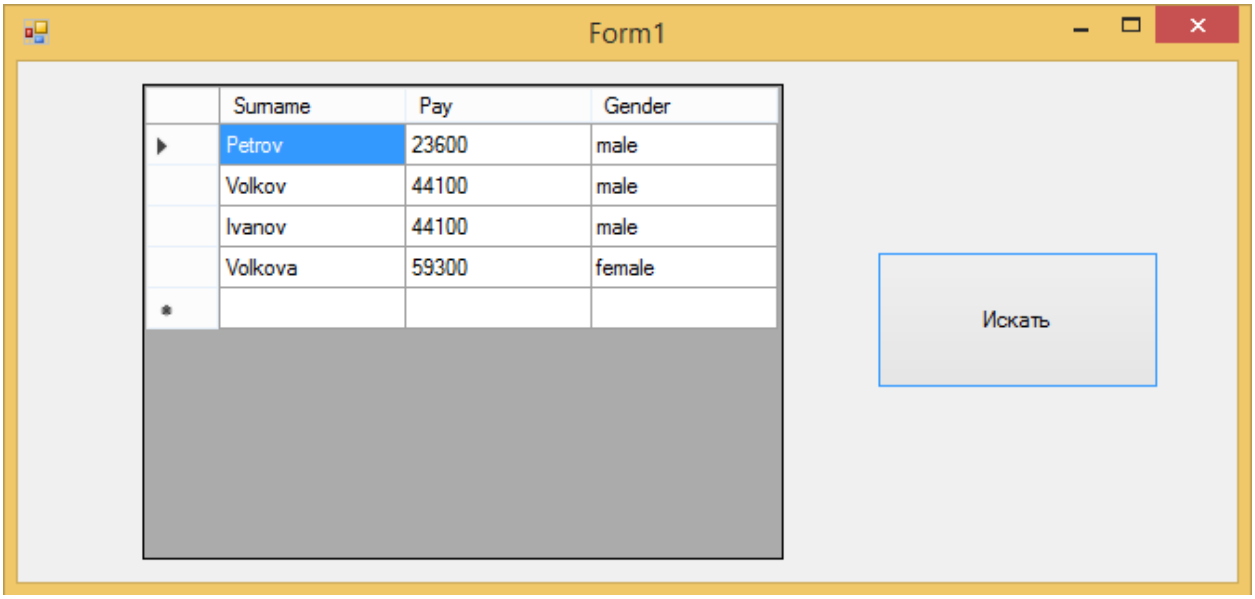

Рисунок 3 – Работа программы со структурой

Индивидуальные задания

1. Имеются данные о сотрудниках фирмы (фамилия, зарплата, пол). Найти фамилию мужчины, имеющего самую высокую зарплату. Если таких сотрудником несколько, то вывести всех, упорядочив список фамилий по возрастанию.

2. Имеются данные о сотрудниках фирмы (фамилия, возраст, отношение к воинской службе). Найти фамилию сотрудника, старшего по возрасту и военнообязанного. Если таких сотрудником несколько, то вывести всех, упорядочив список фамилий по возрастанию.

3. Имеются данные о сотрудниках фирмы (фамилия, зарплата, пол). Найти фамилию женщины, имеющей самую низкую зарплату. Если таких сотрудником несколько, то вывести всех, упорядочив список фамилий по возрастанию.

4. Имеются данные о сотрудниках фирмы (фамилия, возраст, отношение к воинской службе). Найти фамилию сотрудника, младшего по возрасту и невоеннообязанного. Если таких сотрудником несколько, то вывести всех, упорядочив список фамилий по возрастанию.

5. Имеются данные о поездах, проходящих через станцию (номер и назначение поезда, например, 73 Москва-Львов, время прибытия (целочисленные значения часов и минут), время отправления (целочисленные значения часов и минут)). Поезда проходят каждый день. По данному времени определить, какие поезда (номер и назначение поезда) стоят в этот момент на станции.

6. Известна информация о багаже пассажиров (фамилия, количество вещей и общий вес багажа). Найти число пассажиров, имеющих более двух вещей, и вывести фамилии пассажиров.

7. Известна информация о багаже пассажиров (фамилия, количество вещей и общий вес багажа). Выяснить, имеется ли хотя бы один пассажир, багаж которого состоит из одной вещи весом менее 25 кг.

8. Известна информация о багаже пассажиров (фамилия, количество вещей и общий вес багажа). Найти количество пассажиров и их фамилии, вес багажа которых превышает среднее значение багажа всех пассажиров.

9. Известна информация о багаже пассажиров (фамилия, количество вещей и общий вес багажа). Найти количество пассажиров и их фамилии, количество вещей в багаже пассажиров превышает среднее число вещей всех пассажиров.

10. Известны данные о росте учеников класса (фамилия, рост). В начале учебного года поступил новый ученик. Вывести фамилии учеников, рост которых не превышает рост нового ученика.

11. Известны данные о росте учеников класса (фамилия, рост). Данные упорядочены по убыванию значений роста. В начале учебного года поступил новый ученик. Вывести фамилию ученика, после которого следует записать фамилию нового ученика, чтобы упорядоченность не нарушилось.

12. Известны данные о росте учеников класса (фамилия, рост). Данные упорядочены по убыванию значений роста. В начале учебного года поступил новый ученик. Вывести фамилию ученика, рост которого меньше всего отличается от роста нового ученика.

13. Известны данные о росте учеников класса (фамилия, рост). В начале учебного года поступил новый ученик. Вывести фамилии учеников, рост которых превышает рост нового ученика.

14. Известны данные о росте учеников класса (фамилия, рост). Вывести фамилии учеников, рост которых занимает второе и третье место в классе.

15. Известны данные о росте учеников класса (фамилия, рост). Данные упорядочены по убыванию значений роста. В начале учебного года поступил новый ученик. Вывести фамилию ученика, рост которого больше всего отличается от роста нового ученика.

16. Известны данные о трех оценках учеников класса (фамилия, оценки). Вывести фамилии учеников, имеющие лучшую успеваемость.

17. Известны данные о трех оценках учеников класса (фамилия, оценки). Вывести фамилии учеников, имеющие худшую успеваемость.

18. Известны данные о трех оценках учеников класса (фамилия, оценки). Вывести фамилии учеников, имеющие сумму оценок, занимающую предпоследнее место в классе.

19. Известны данные о трех оценках учеников класса (фамилия, оценки). Вывести фамилии учеников, имеющие сумму оценок, занимающую второе место в классе.

20. Известны данные о трех оценках учеников класса (фамилия, оценки). Вывести фамилии учеников, имеющие сумму оценок, превышающую средний балл в классе..

21. Имеются данные о днях месяца (температура воздуха, количество осадков). Определить количество осадков, выпавших в виде снега (считать, что идет снег, если температура меньше 0).

22. Имеются данные о днях месяца (температура воздуха, количество осадков). Определить количество осадков, выпавших в виде дождя (считать, что идет дождь, если температура больше 0).

23. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Найти среднюю стоимость легковых автомобилей и вывести марки легковых автомобилей, стоимость которых превышает среднюю стоимость.

24. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Найти среднюю стоимость легковых автомобилей и вывести марки легковых автомобилей, стоимость которых превышает среднюю стоимость.

25. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Вывести марки легковых автомобилей, имеющих самую высокую стоимость.

26. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Вывести марки грузовых автомобилей, имеющих самую низкую стоимость.

27. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Был куплен новый грузовик. Определить, является ли стоимость нового грузовика больше средней стоимости грузовиков.

28. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Данные упорядочены по убыванию стоимостей. Был куплен новый грузовик. Определить стоимость грузовика, наиболее близкую стоимости нового грузовика.

29. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Данные упорядочены по убыванию стоимостей. Был куплен новый легковой автомобиль. Добавить данные о новом автомобиле, чтобы упорядоченность не нарушилась.

30. В фирме имеются данные о легковых и грузовых автомобилях. (марка, тип (грузовой или легковой), стоимость). Данные упорядочены по убыванию стоимостей. Был куплен новый грузовик. Добавить данные о новом автомобиле, чтобы упорядоченность не нарушилась.

Контрольные вопросы.

1. В чём различия структур и классов?

2. Назовите допустимые модификаторы структур?
3. Что такое интерфейсы структур?
4. Почему в структурах отсутствует финализатор?
5. Объясните особенности копирования структур?

Лабораторная работа №11: «Обработка исключительных ситуаций на языке C#»

Цель работы — Основные понятия. Варианты с использования конструкции try...catch...finally.

Основные понятия

Иногда при выполнении программы возникают ошибки, которые трудно предусмотреть или предвидеть, а иногда и вовсе невозможно. Например, при передачи файла по сети может неожиданно оборваться сетевое подключение. такие ситуации называются **исключениями**. Язык C# предоставляет разработчикам возможности для обработки таких ситуаций. Для этого в C# предназначена конструкция **try...catch...finally**. При возникновении исключения среда CLR ищет блок catch, который может обработать данное исключение. Если такого блока не найдено, то пользователю отображается сообщение о необработанном исключении, а дальнейшее выполнение программы останавливается. И чтобы подобной остановки не произошло, и надо использовать блок try..catch. Пример:

```
static void Main(string[] args)
{
   int[] a = new int[4];    try
   {
      a[5] = 4; // тут возникнет исключение, так как у нас в массиве только 4 элемента
      Console.WriteLine("Завершение блока try");
   }
   catch (Exception ex)
   {
      Console.WriteLine("Ошибка: " + ex.Message);
   }
   finally    {
      Console.WriteLine("Блок finally");
   }
   Console.ReadLine();
}
```

При использовании блока try...catch..finally вначале выполняются все инструкции между операторами try и catch. Если между этими

операторами вдруг возникает исключение, то обычный порядок выполнения останавливается и переходит к инструкции catch. В данном случае у нас явно возникнет исключение в блоке try, так как мы пытаемся присвоить значение шестому элементу массива в то время, как в массиве всего 4 элемента. И дойдя до строки a[5] = 4;, выполнение программы остановится и перейдет к блоку catch

Инструкция catch имеет следующий синтаксис: catch (тип_исключения имя_переменной). В нашем случае объявляется переменная ex, которая имеет тип Exception. Но если возникшее исключение не является исключением типа, указанного в инструкции catch, то оно не обрабатывается, а программа просто зависает или выбрасывает сообщение об ошибке.

Однако так как тип Exception является базовым классом для всех исключений, то выражение catch (Exception ex) будет обрабатывать практически все исключения. Вся обработка исключения в нашем случае сводится к выводу на консоль сообщении об исключении, которое в свойстве message класса Exception.

Далее в любом случае выполняется блок finally. Однако этот блок необязательный, и его можно при обработке исключений опускать. Если же в ходе программы исключений не возникнет, то программа не будет выполнять блок catch, сразу перейдет к блоку finally, если он имеется.

Обработка нескольких исключений

При необходимости мы можем разграничить обработку различных типов исключений, включив дополнительные блоки catch:

```
static void Main(string[] args)
{    try    {

    }
        catch (FileNotFoundException e)
        {
            // Обработка исключения, возникшего при отсутствии файла      }
        catch (IOException e)
```

```
        {
            // Обработка исключений ввода-вывода
        }
    Console.ReadLine();
}
```

Если у нас возникает исключение определенного типа, то оно переходит к соответствующему блоку catch.

При этом более частные исключения следует помещать в начале, и только потом более общие классы исключений. Например, сначала обрабатывается исключение IOException, и только потом Exception (так как IOException наследуется от класса Exception).

Оператор throw

Чтобы сообщить о выполнении исключительных ситуаций в программе, можно использовать оператор throw. То есть с помощью этого оператора мы сами можем создать исключение и вызвать его в процессе выполнения. Например, в нашей программе происходит ввод строки, и мы хотим, чтобы, если длина строки будет больше 6 символов, возникало исключение:

```
static void Main(string[] args)
{   try
    {
            string message = Console.ReadLine();                if
(message.Length > 6)
            {               throw new Exception("Длина строки больше 6
символов");
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Ошибка: " + e.Message);
        }
    Console.ReadLine();
}
```

Обработка исключений и условные конструкции

Ряд исключительных ситуаций может быть предвиден разработчиком. Например, пусть программа предусматривает ввод числа и вывод его квадрата:

```
static void Main(string[] args)
{
   Console.WriteLine("Введите число");                int x = Int32.Parse(Console.ReadLine());

   x *= x;
   Console.WriteLine("Квадрат числа: " + x);
      Console.Read();
}
```

Если пользователь введет не число, а строку, какие-то другие символы, то программа выпадет в ошибку. С одной стороны, здесь как раз та ситуация, когда можно применить блок try..catch, чтобы обработать возможную ошибку. Однако гораздо оптимальнее было бы проверить допустимость преобразования:

```
static void Main(string[] args)
{
   Console.WriteLine("Введите число");     int x;
   string input = Console.ReadLine();      if (Int32.TryParse(input, out x))
   {
      x *= x;
      Console.WriteLine("Квадрат числа: " + x);
   }     else
   {
      Console.WriteLine("Некорректный ввод");
   }
      Console.Read();
}
```

Метод Int32.TryParse() возвращает true, если преобразование можно осуществить, и false - если нельзя. При допустимости преобразования переменная x будет содержать введенное число. Так,

не используя try...catch можно обработать возможную исключительную ситуацию.

С точки зрения производительности использование блоков try..catch более накладно, чем применение условных конструкций. Поэтому по возможности вместо try..catch лучше использовать условные конструкции на проверку исключительных ситуаций.

Фильтры исключений

В C# 6.0 (Visual Studio 2015) была добавлена такая функциональность, как фильтры исключений. Они позволяют обрабатывать исключения в зависимости от определенных условий:

```
int x = 1; int y = 0;
try {     int result = x / y;
}
catch(Exception ex) when (y==0)
{
   Console.WriteLine("y не должен быть равен 0");
}
catch(Exception ex)
{
   Console.WriteLine(ex.Message); }
```

В данном случае будет выброшено исключение, так как y=0. Здесь два блока catch, но поскольку для первого блока указано условие с помощью ключевого слова when, то сработает первый блок catch. Если бы y не было бы равно 0, то сработал бы второй блок catch.

ЗАДАНИЕ

Общее задание для всех вариантов: в программы из ЛР№4 и ЛР №6 добавить обработку исключительных ситуаций.

Лабораторная работа №12: «Работа с файлами в языке C#»

Цель работы ─ изучение и приобретение навыков ввода и вывода файлов, выполнения действий над разными форматами файлов.

Основные понятия

Файл – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске). Файл имеет имя и расширение. Расширение позволяет идентифицировать, какие данные и в каком формате хранятся в файле.

Под работой с файлами подразумевается:

- создание файлов;
- удаление файлов;
- чтение данных;
- запись данных;
- изменение параметров файла (имя, расширение…);
- другое.

В C# есть пространство имен **System.IO**, в котором реализованы все необходимые нам классы для работы с файлами.

Чтобы подключить это пространство имен, необходимо в самом начале программы добавить строку:

using System.IO;

Для использования кодировок еще нужно добавить пространство:

using System.Text;

Классы

Многие типы из пространства имен System.IO сосредоточены на программных манипуляциях физическими каталогами и файлами. Дополнительные типы предоставляют поддержку чтения и записи данных в строковые буферы, а также области памяти. Ниже кратко описаны основные (неабстрактные) классы, которые дают понятие о функциональности System.IO:

BinaryReader, BinaryWriter

Эти классы позволяют сохранять и извлекать элементарные типы данных (целочисленные, булевские, строковые и т.п.) в двоичном виде;

BufferedStream

Этот класс предоставляет временное хранилище для потока байтов, которые могут затем быть перенесены в постоянные хранилища;

Directory, DirectoryInfo

Эти классы используются для манипуляций структурой каталогов машины. Тип Directory представляет функциональность, используя статические члены.

Тип DirectoryInfo обеспечивает аналогичную функциональность через действительную объектную ссылку;

DriveInfo

Этот класс предоставляет детальную информацию относительно дисковых устройств, используемых данной машиной;

File, FileInfo

Эти классы служат для манипуляций множеством файлов данной машины. Тип File представляет функциональность через статические члены. Тип FileInfo обеспечивает аналогичную функциональность через действительную объектную ссылку;

FileStream

Этот класс обеспечивает произвольный доступ к файлу (т.е. возможности поиска) с данными, представленными в виде потока байт;

FileSystemWatcher

Этот класс позволяет отслеживать модификации внешних файлов в определенном каталоге;

MemoryStream

Этот класс обеспечивает произвольный доступ к данным, хранящимся в памяти, а не в физическом файле;

Path

Этот класс выполняет операции над типами System.String, содержащими информацию о пути к файлу или каталогу в независимой от платформы манере StreamWriter, StreamReader;

Эти классы используются для хранения (и извлечения) текстовой информации из файла. Эти классы не поддерживают произвольного доступа к файлу.

StringWriter, StringReader

Подобно классам StreamWriter/StreamReader, эти классы также работают с текстовой информацией. Однако лежащим в основе хранилищем является строковый буфер, а не физический файл. По названию классов видно, что первый используется для работы с файлами в режиме записи, второй для работы в режиме чтения. Они интуитивно понятны, так как все их методы явно отражают действия.

В дополнение к этим конкретным типам классов в System.IO определено несколько перечислений, а также набор абстрактных классов (т.е. Stream,

TextReader и TextWriter), которые определяют разделяемый полиморфный интерфейс для всех наследников.

Как не трудно было заметить в приведенном выше списке, для представления файлов и папок используются по два класса. Какой из них применять — во многом зависит от того, сколько раз требуется получить доступ к данной папке или файлу.

Методы

Основными классами для выполнения действий с файлами предназначены File и FileInfo. С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Класс FileInfo позволяет получать подробности относительно существующих файлов на жестком диске (т.е. время создания, размер и атрибуты) и предназначен для создания, копирования, перемещения и удаления файлов. Вдобавок к набору функциональности, унаследованной от FileSystemInfo, есть некоторые члены, уникальные для класса FileInfo.

Таблица 3 _Некоторые полезные методы и свойства класса ***FileInfo***

WriteAllText()	Создает новый файл, записывает в него содержимое и затем закрывает файл. Если целевой файл уже существует, он будет переопределен.
AppendAllText()	Открывает файл, добавляет в него указанную строку и затем закрывает файл. Если файл не существует, этот метод создает файл, записывает в него указанную строку и затем закрывает файл.
CopyTo(path)	Копирует файл в новое место по указанному пути path
Create()	Создает файл и возвращает объект FileStream для взаимодействия с вновь созданным файлом
CreateText()	Создает объект StreamWriter, записывающий новый текстовый файл
Delete()	Удаляет файл , к которому привязан экземпляр FileInfo
Directory	Получает родительский каталог в виде объекта DirectoryInfo
DirectoryName	Получает полный путь к родительскому каталогу
Exists	Указывает, существует ли файл
Extension	Получает расширение файла

Length	Получает размер текущего файла или каталога
MoveTo(destFileName)	Перемещает файл в новое место
Name	Получает имя файла
FullName	Получает полное имя файла
Open()	Открывает файл с различными привилегиями чтения/записи и совместного доступа
OpenRead()	Создает доступный только для чтения объект FileStream
OpenText()	Создает объект StreamReader и читает из существующего текстового файла
OpenWrite()	Создает доступный только для записи объект FileStream
ReadAllText()	Читает файл целиком
ReadAllLines()	Возвращает массив строк, т.е. читает файл построчно

Обратите внимание, что большинство методов класса FileInfo возвращают специфический объект ввода-вывода (т.е. FileStream и StreamWriter), который позволяет начать чтение и запись данных в ассоциированный файл в разнообразных форматах.

Класс ***File*** реализует похожую функциональность с помощью статических методов:

Таблица 4 _Н статические методы

Copy()	копирует файл в новое место
Create()	создает файл
Delete()	удаляет файл
Move	перемещает файл в новое место
Exists(file)	определяет, существует ли файл

Теперь рассмотрим каждый метод подробнее:

WriteAllText()

```
static void Main(string[] args)
{
  File. WriteAllText("D:\\new_file.txt", "текст");
}
```

Создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле.

AppendAllText()

```
static void Main(string[] args)
{
  File.AppendAllText("D:\\new_file.txt", "текст метода AppendAllText ()"); //допишет текст в конец файла
}
```

Данный метод работает, как и метод WriteAllText() за исключением того, что новый текст дописывается в конец файла, а не переписывает всё, что было в файле.

CopyTo()

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
  fileInf.CopyTo(newPath, true);
  // альтернатива с помощью класса File
  // File.Copy(path, newPath, true);
}
```

Метод CopyTo класса FileInfo принимает два параметра: путь, по которому файл будет копироваться, и булевое значение, которое указывает, надо ли при копировании перезаписывать файл (если true, как в случае выше, файл при копировании перезаписывается). Если же в качестве последнего параметра передать значение false, то если такой файл уже существует, приложение выдаст ошибку.

Метод Copy класса File принимает три параметра: путь к исходному файлу, путь, по которому файл будет копироваться, и булевое значение, указывающее, будет ли файл перезаписываться.

Create()

```
static void Main(string[] args)
{
  File.Create("D:\\new_file.txt");
}
```

Для создания пустого файла, в классе File есть метод Create(). Он принимает один аргумент – путь. Выше приведен пример создания пустого текстового файла new_file.txt на диске D.

CreateText()

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
if (text.Exists)
text.CreateText();
```

Delete()

```
static void Main(string[] args)
{
  File.Delete("D:\\test.txt"); //удаление файла
}
```

Метод Delete() удаляет файл по указаному пути.

Directory

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
if (text.Exists)
 MessageBox.Show(text.Directory.FullName);
// результат: C:\Users\ Desktop\
```

DirectoryName

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   Console.WriteLine(fileInf.DirectoryName);
}
// результат: C:\apache\
```

Exists

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
MessageBox.Show("+")
}
// результат: +, в случае если файл существует
```

Extension

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
if (text.Exists)
```

```
MessageBox.Show(text.Extension);
// результат: .txt
```

Length

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
Console.WriteLine("Размер: {0}", fileInf.Length);
}
// результат: 0, поскольку файл пуст
```

MoveTo()

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
  fileInf.MoveTo(newPath);
  // альтернатива с помощью класса File
  // File.Move(path, newPath);
}
```

Name

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   Console.WriteLine("Имя файла: {0}", fileInf.Name);
}
// результат: hta.txt
```

FullName

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   Console.WriteLine("Имя файла: {0}", fileInf.FullName);
}
// результат: C:\apache\hta.txt
```

Open()

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
 if (text.Exists)
text.Open(FileMode.Open);
// результат: открывает файл
```

OpenRead()

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
 if (text.Exists)
text.OpenRead ();
// результат: открывает файл для чтения
```

OpenText()

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
 if (text.Exists)
text.OpenText();
// результат: открывает для чтения файл (текст в кодировке UTF-8).
```

OpenWrite()

```
string main_file = @"C:\Users\ Desktop\SWSU.txt";
FileInfo text = new FileInfo(main_file);
 if (text.Exists)
text.OpenWrite();
// результат: открывает файл для записи
```

Пример программы с использованием класса BinaryWriter

Условие задачи

В соревнованиях по прыжкам в высоту принимали участие спортсмены нескольких стран. О каждом участнике известны следующие сведения: фамилия, возраст, страна, спортивный разряд, сведения о дисквалификации

Схема алгоритма программы

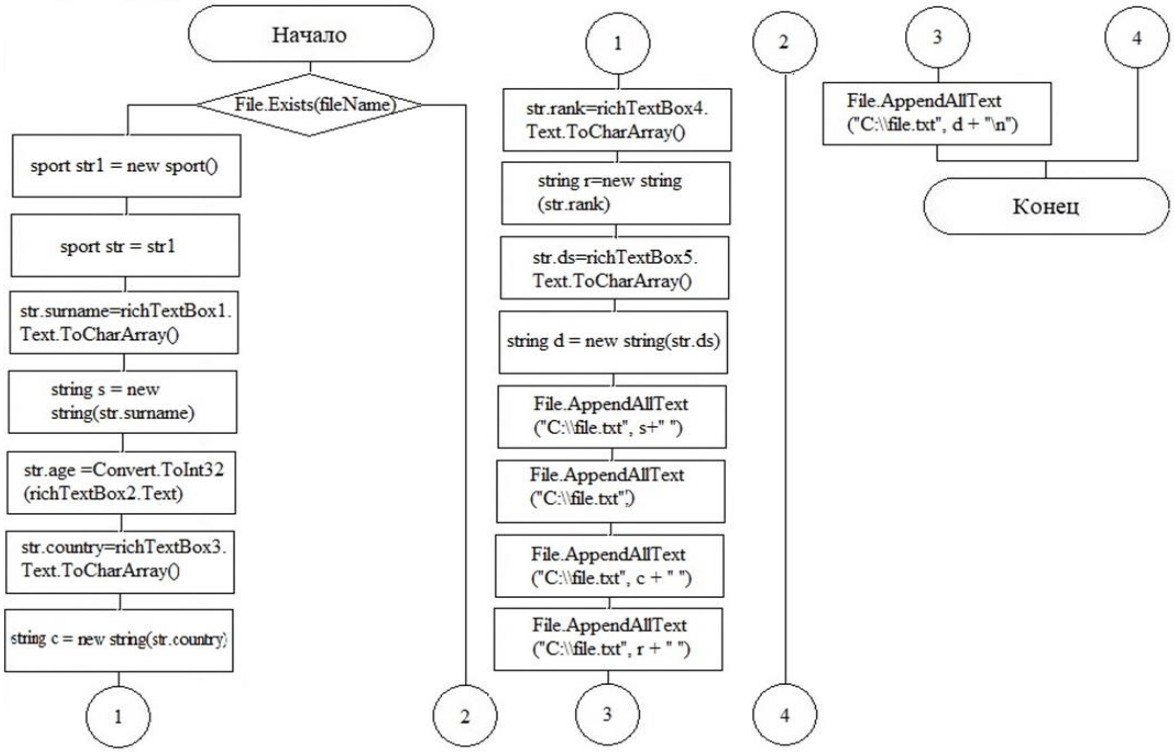

Рисунок 4. Алгоритм примера программы по прыжкам в высоту

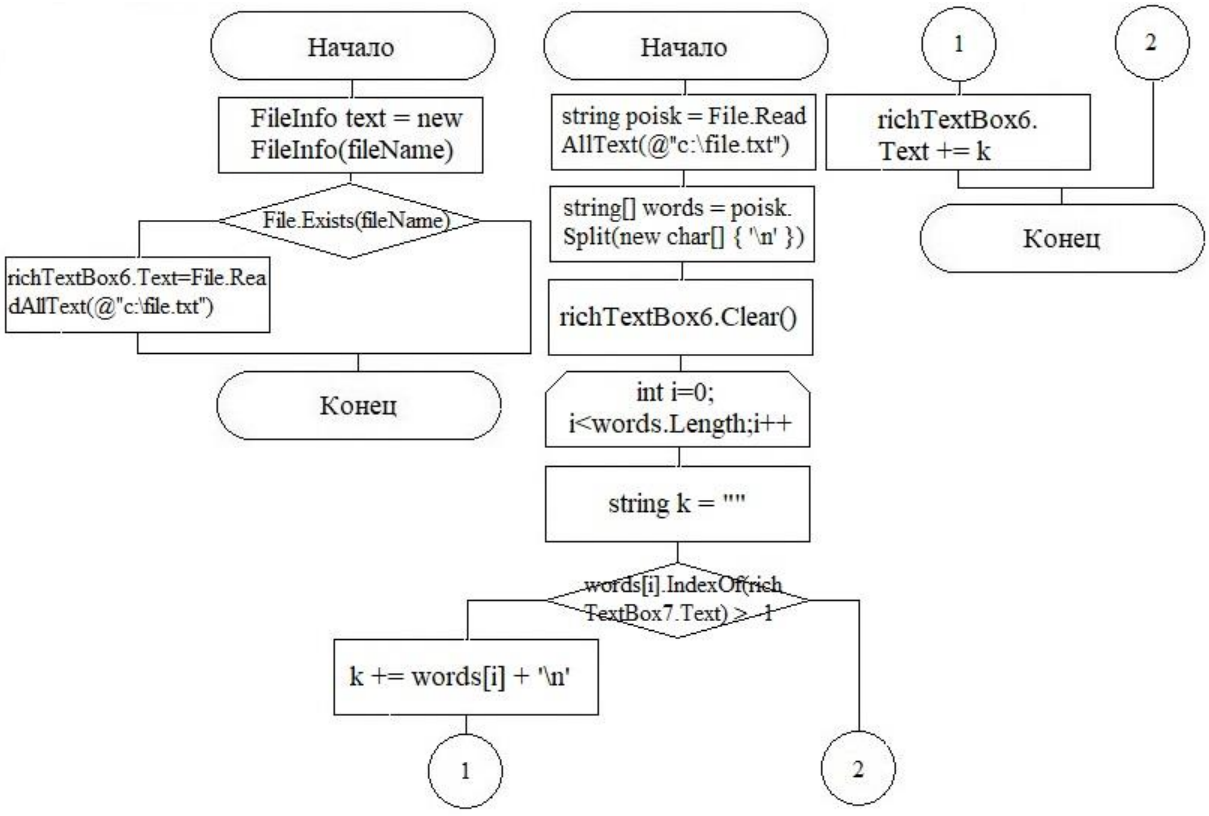

Рисунок 5. Алгоритм продолжения программы по прыжкам в высоту

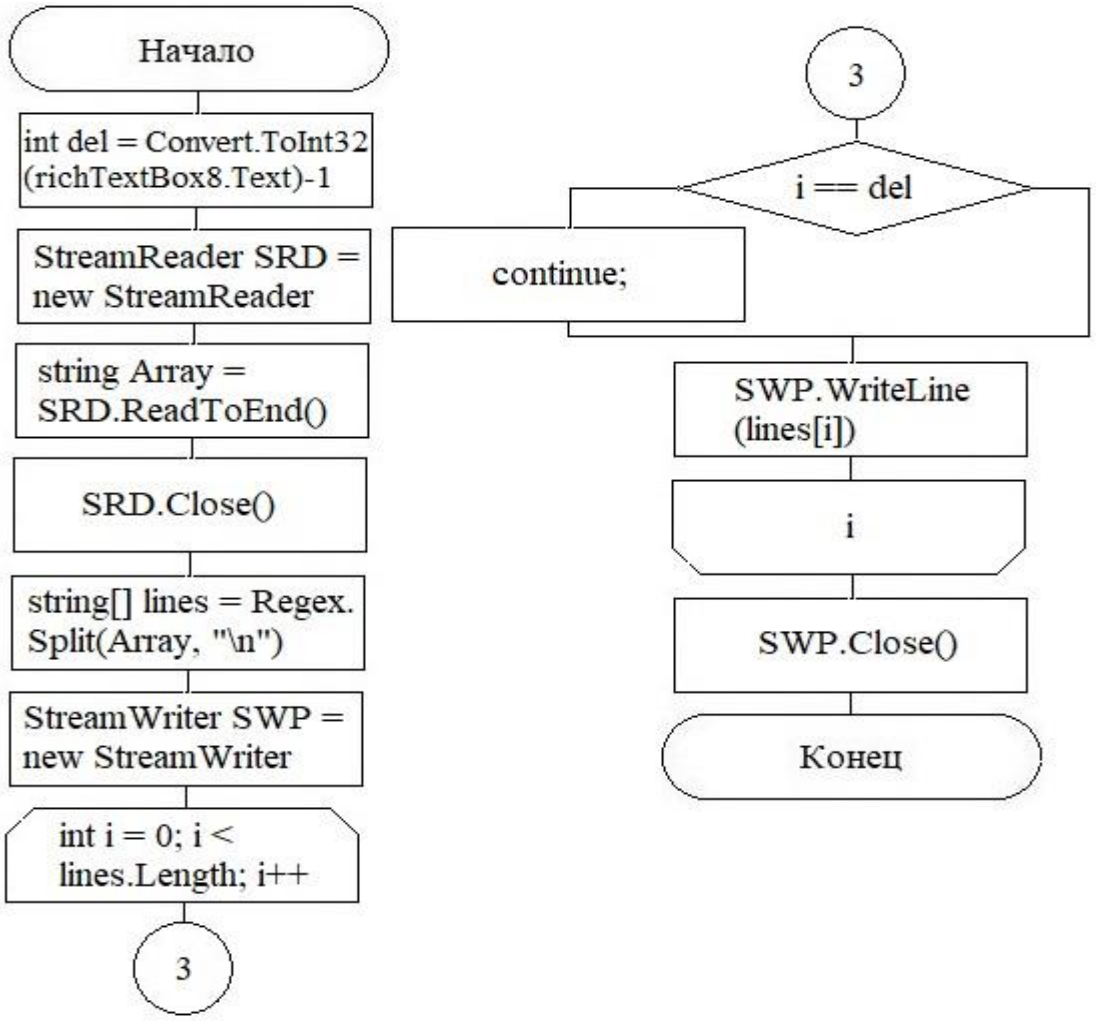

Рисунок 6. Алгоритм продолжения программы по прыжкам в высоту

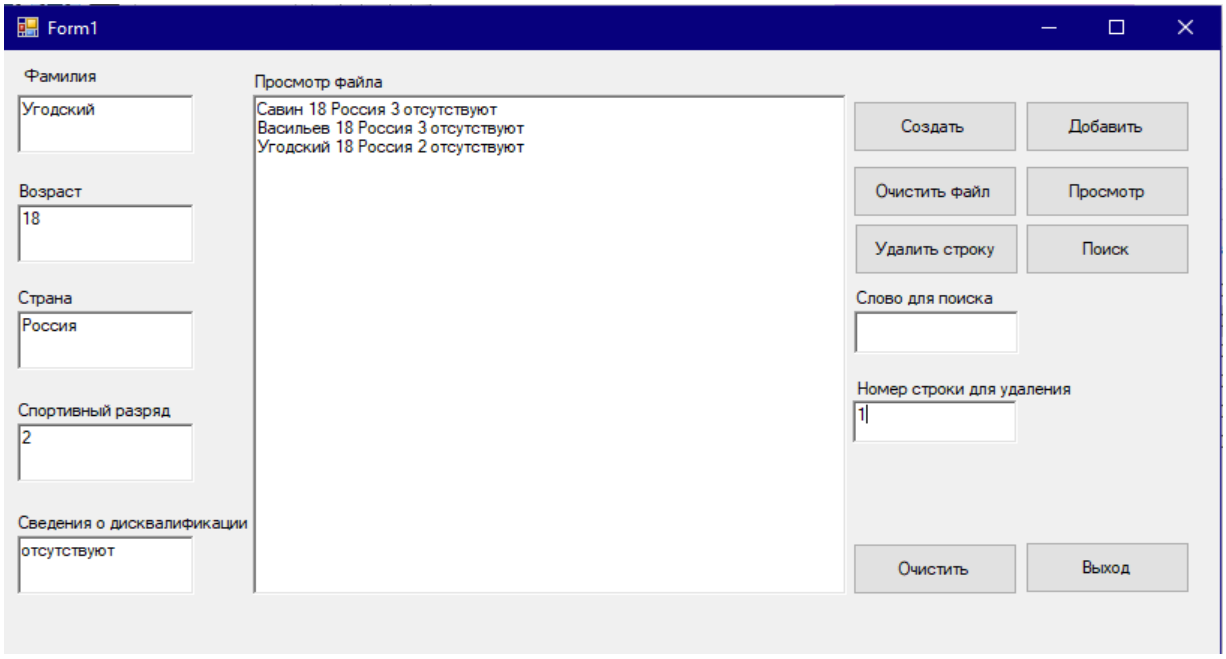

Рисунок 7. Интерфейс программы записи в файл

Код программы:

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp12
{
public partial class Form1 : Form
{

struct sport
{
public char[] surname;
public int age;
public char[] country;
public char[] rank;
public char[] ds;
}

public Form1()
{
InitializeComponent();
}

const string fileName = "file.txt";

private void button1_Click(object sender, EventArgs e)
{
using (BinaryWriter writer = new BinaryWriter(File.Open(fileName, FileMode.Create)))
{
```

```
        }
        }

        private void button2_Click(object sender, EventArgs e)
        {
        if(File.Exists(fileName))
        {
        using (BinaryWriter writer = new BinaryWriter(File.Open(fileName,
FileMode.Append)))
        {
        sport str1 = new sport();
        sport str = str1;
        str.surname =richTextBox1.Text.ToCharArray();
        string s = new string(str.surname);
        str.age =Convert.ToInt32( richTextBox2.Text );
        str.country = richTextBox3.Text.ToCharArray();
        string c = new string(str.country);
        str.rank = richTextBox4.Text.ToCharArray();
        string r = new string(str.rank);
        str.ds = richTextBox5.Text.ToCharArray();
        string d = new string(str.ds);

        File.AppendAllText("C:\\file.txt", s+"");
        File.AppendAllText("C:\\file.txt", Convert.ToString(str.age + ""));
        File.AppendAllText("C:\\file.txt", c + "");
        File.AppendAllText("C:\\file.txt", r + "");
        File.AppendAllText("C:\\file.txt", d + "\n");
        }
        }
        }
        private void button4_Click(object sender, EventArgs e)
        {
        FileInfo text = new FileInfo(fileName);
        if (File.Exists(fileName))
        {
        richTextBox6.Text = File.ReadAllText(@"c:\file.txt");
        }
        }

        private void button3_Click(object sender, EventArgs e)
        {
```

```
        DialogResult dialogResult = MessageBox.Show("Вы точно хотите очистить файл?", "Очистить файл?", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
        File.WriteAllText(@"c:\file.txt", string.Empty);
        MessageBox.Show("Файл очищен");
        }
        else if (dialogResult == DialogResult.No)
        {
        MessageBox.Show("А зачем ты тогда сюда жмякал????");
        }

        }

        private void button5_Click(object sender, EventArgs e)
        {
        string poisk = File.ReadAllText(@"c:\file.txt");
        string[] words = poisk.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);
        richTextBox6.Clear();
        for (int i=0; i<words.Length;i++)
        {
        string k = "";
        if (words[i].IndexOf(richTextBox7.Text) > -1)
        {
        k += words[i] + '\n';
        richTextBox6.Text += k;
        continue;
        }
        }
        }

        private void button7_Click(object sender, EventArgs e)
        {
        Application.Exit();
        }

        private void button6_Click(object sender, EventArgs e)
        {
        richTextBox1.Clear();
        richTextBox2.Clear();
        richTextBox3.Clear();
```

```
richTextBox4.Clear();
richTextBox5.Clear();
richTextBox6.Clear();
richTextBox7.Clear();
richTextBox8.Clear();
}
private void button8_Click(object sender, EventArgs e)
{
int del = Convert.ToInt32(richTextBox8.Text)-1; // Номер индекса строки которую надо удалить. Первая строку будет под 1, а не под 0.
StreamReader SRD = new StreamReader(@"c:\file.txt", Encoding.GetEncoding(1251));
string Array = SRD.ReadToEnd();
SRD.Close();
string[] lines = Regex.Split(Array, "\n");
StreamWriter SWP = new StreamWriter(@"c:\file.txt", false, Encoding.GetEncoding(1251));
for (int i = 0; i < lines.Length; i++)
{
if (i == del)
continue;
SWP.WriteLine(lines[i]);
}
SWP.Close();
}
}
}
```

Результат выполнения программы

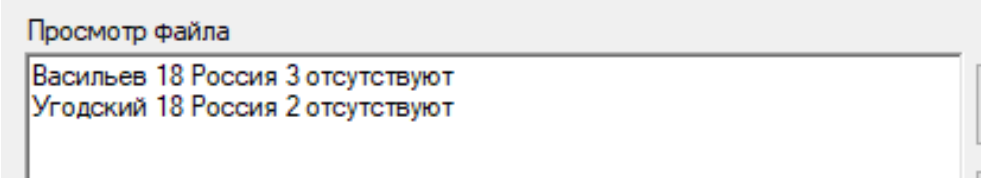

Рисунок 8. Результат выполнения программы

Задания №1к лабораторной работе

Написать программу, которая выполняет с файлом следующие действия:

— создание файла;
— добавление записи в файл;
— просмотр содержимого файла;
— удаление записи из файла;
— поиск записи в файле.

1. Имеются результаты медицинского обследования группы детей, в которых указаны: фамилия, адрес, возраст, вес, рост и наличие прививок по возрасту.

2. Есть список сотрудников с указанием фамилии, адреса, даты рождения, профессии, общего стажа работы, стажа работы в данной организации, оклада согласно штатному расписанию.

3. В библиотеке хранятся сведения о книгах: название книги, автор, год выпуска, стоимость, издательство, как часто книга пользовалась спросом у читателей.

4. В кинотеатрах города демонстрируются фильмы. Известны названия кинотеатров, названия фильмов, киностудия, режиссёр, дата первого показа и дата последнего показа.

5. В соревнованиях по легкой атлетике принимали участие спортсмены из разных городов. О каждом участнике известны следующие сведения: фамилия, возраст, город, спортивный разряд, сведения о дисквалификации.

6. В банке данных хранятся сведения о городах Центрально-Чернозёмного района: название города, год основания, количество жителей, занимаемая площадь, количество экологически вредных предприятий.

7. В администрации хранятся сведения о фирмах: название, дата регистрации, форма собственности, № счёта, сумма доходов за предыдущий год, количество сотрудников.

8. В отделе кадров фирмы хранятся сведения о рабочих: табельный номер, пол, поступление на работу, разряд по профессии, сведения о нарушениях трудовой дисциплине.

9. На бирже труда хранятся сведения о безработных: фамилия, образование, год рождения, количество иждивенцев, дата обращения на биржу, профессия, последнее место работы, размер заработной платы.

10. В деканате хранится список студентов с указанием № группы, фамилии и экзаменационных оценок по 5 предметам, полученных за последнюю сессию.

11. В областном центре по сельскому хозяйству имеются следующие сведения: название района, размер площадей засеваемых зерновыми, урожайность с 1 га, сведения о транспортном парке.

12. В детском саду хранятся сведения о детях: фамилия, имя, адрес, год рождения, перенесённые болезни.

13. В КВН принимают участие команды из разных городов. О каждой команде известно: название, город, количество участников, капитан, количество проигрышей и побед.

14. В адресно-справочном бюро хранится информация о жителях города: Ф.И.О., год рождения, адрес прописки, адрес жительства, мобильный телефон.

15. В поликлинике хранятся сведения о пациентах: Ф.И.О., год рождения, прописка, место работы (учёбы), перенесённые болезни, стоит ли человек на учёте (по какой болезни и у какого врача).

16. В базе данных роддома хранится информация о детях: Ф.И.О., пол ребёнка, его год рождения, вес, рост, наличие врождённых болезней.

17. В отделении почты хранятся сведения о подписчиках на газеты и журналы: Ф.И.О., адрес, названия газет и журналов, временной период подписки.

18. Есть список планет солнечной системы: номер планеты по удалению от Солнца, название планеты, объём, диаметр, удалённость от Земли.

19. На складе имеется информация о хранящемся товаре: код товара, наименование товара, дата выпуска, цена, количество.

20. Даны результаты лучших спортивных достижений студентов университета: название вида спорта, Ф.И.О. и возраст рекордсмена, дата установления рекорда, описание рекорда.

21. В базе данных университета содержится информация о студентах: Ф.И.О., пол, год рождения, адрес, личный номер студента.

22. В автосалоне хранится информация о машинах: марка, год выпуска, страна, технические характеристики, цена.

23. На автовокзале имеется информация о расписании автобусов: номер рейса, пункт назначения, время отправления, время прибытия, марка автобуса, количество мест.

24. На вокзале имеется информация о расписании поездов: номер рейса, пункт назначения, время отправления, время прибытия, тип поезда, количество мест.

25. В аэропорту имеется информация о расписании самолетов: номер рейса, пункт назначения, время отправления, время прибытия, тип самолета, количество мест.

26. В турагентстве хранится информация о путевках: страна, город, количество дней, уровень сервиса, стоимость.

27. В видеотеке хранится информация о фильмах: название, год выпуска, жанр, режиссер, студия.

28. В электронном деканате университета хранится информация: Ф.И.О., баллы за посещаемость, баллы за успеваемость, премиальные баллы деканата, премиальные баллы преподавателя, баллы, полученные на экзамене, сумма баллов.

29. В речном порту имеется информация о расписании речных судов: номер рейса, пункт назначения, время отправления, время прибытия, тип судна, количество мест.

30. В фитнес-клубе хранится информация о людях, которые приобрели абонементы: Ф.И.О., пол, год рождения, адрес, паспортные данные, мобильный телефон, временной период действия абонемента.

Задание №2 к лабораторной работе

1. Даны два текстовых файла. Добавить в начало первого файла содержимое второго файла.

2. Даны два текстовых файла. Добавить в конец первого файла содержимое второго файла.

3. Дано целое число k и текстовый файл. Вставить пустую строку перед строкой файла с номером k. Если строки с таким номером нет, то поставить пустую строку в начало файла.

4. Дано целое число k и текстовый файл. Вставить пустую строку после строки файла с номером k. Если строки с таким номером нет, то поставить 2 пустые строки в начало файла.

5. Дан текстовый файл. Продублировать в нем все пустые строки. Если пустых строк нет, то после каждой строки вставить по одной пустой строке.

6. Дана строка s и текстовый файл. Заменить в файле все пустые строки на строку s.

7. Дан непустой текстовый файл. Удалить из него первую строку.

8. Дан непустой текстовый файл. Удалить из него последнюю строку.

9. Дано целое число k и текстовый файл. Удалить из файла строку с номером k. Если строки с таким номером нет, то оставить файл без изменений.

10. Дан текстовый файл. Удалить из него все пустые строки.

11. Даны два текстовых файла. Добавить в конец каждой строки первого файла соответствующую строку второго файла. Если второй файл короче первого, то оставшиеся строки первого файла не изменять.

12. Дано целое число k и текстовый файл. Удалить из каждой строки файла первые k символов (если длина строки меньше k, то удалить из нее все символы).

13. Дан текстовый файл. Заменить в нем все прописные русские буквы на строчные, а все строчные — на прописные.

14. Дан текстовый файл. Заменить в нем все подряд идущие пробелы на один пробел.

15. Дан текстовый файл, содержащий более трех строк. Удалить из него последние три строки.

16. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Удалить из файла последние k строк.

17. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Создать новый текстовый файл, содержащий k последних строк исходного файла.

18. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Создать новый текстовый файл, содержащий k первых строк исходного файла

19. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Создать новый текстовый файл, содержащий k строк, начиная со второй строки исходного файла

20. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Создать новый текстовый файл, содержащий k строк, начиная с третьей строки исходного файла

21. Дан текстовый файл. Вывести первое слово текста наибольшей длины. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки.

22. Дан текстовый файл. Вывести последнее слово текста наибольшей длины. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки.

23. Дан текстовый файл. Вывести первое слово текста наименьшей длины. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки.

24. Дан текстовый файл. Вывести последнее слово текста наименьшей длины. Словом считать набор символов, не содержащий пробелов и ограниченный пробелами или началом/концом строки.

25. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Удалить из файла k строк, начиная со второй строки файла.

26. Дано целое число k (0 < k <= 10) и текстовый файл, содержащий более k строк. Удалить из файла k строк, начиная с третьей строки файла.

27. Дан символ c — прописная (заглавная) русская буква и текстовый файл. Создать строковый файл и записать в него все слова из исходного файла, начинающиеся на эту букву (прописную или строчную). Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки. Если исходный файл не содержит подходящих слов, то оставить результирующий файл пустым.

28. Дан символ c — строчная (маленькая) русская буква и текстовый файл. Создать строковый файл и записать в него все слова из исходного файла, содержащие хотя бы одну букву c (прописную или строчную). Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки. Если исходный файл не содержит подходящих слов, то оставить результирующий файл пустым.

29. Дано целое число k и текстовый файл. Создать строковый файл и записать в него все слова длины k из исходного файла. Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки. Если исходный файл не содержит слов длины k, то оставить результирующий файл пустым.

30. Дан символ c — строчная (маленькая) русская буква и текстовый файл. Создать строковый файл и записать в него все слова из исходного файла исключив слова, содержащие хотя бы одну букву c (прописную или строчную). Словом считать набор символов, не содержащий пробелов, знаков препинания и ограниченный пробелами, знаками препинания или началом/концом строки. Если исходный файл не содержит подходящих слов, то оставить результирующий файл пустым.

Список используемых источников

1. Белов В.Г. Основы программирования на языке C++ Builder [Текст]: учеб. пособие / В.Г. Белов, Т.М. Белова; Юго-Зап. гос. ун-т. – Курск, 2015. – 160 с.
2. Белов В.Г. Основы программирования на языке C++ Builder [Электронный ресурс]: учеб. пособие / В.Г. Белов, Т.М. Белова; ЮгоЗап. гос. ун-т. – Курск, 2015. – 160 с.
3. Архангельский, А.Я. Программирование в C++ Builder [Текст] /
4. А.Я. Архангельский. – М.: Изд-во БИНОМ, 2010. – 1304 с.
5. Дэвид Р. Мюссер. C++ и STL. Справочное руководство [Текст] / Дэвид Р. Мюссер, Жилмер Дж. Дердж, Атул Сейни. – М.: Вильямс, 2010. – 432 с.
6. Культин, Н. C++ Builder [Текст] / Н. Культин. – СПб.: БХВПетербург, 2012. – 464 с.
7. Лафоре, Р. Объектно-ориентированное программирование в C++ [Текст] / Р. Лафоре. – СПб.: ПИТЕР, 2013. – 924 с.
8. Прата, С. Язык программирования C++. Лекции и упражнения [Текст] / С. Прата. – М.: Вильямс, 2012. – 1244 с.
9. http://mycsharp.ru/post/21/2013_06_12_rabota_s_fajlami_v_si-sharp_klassy_streamreader_i_streamwriter.html
10. https://metanit.com/sharp/tutorial/5.3.php
11. https://professorweb.ru/my/csharp/thread_and_files/level3/3_2.php