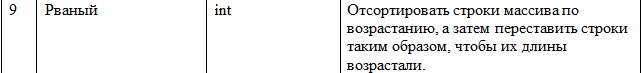
**Отчёт по лабораторной работе №6**

**Постановка задачи вариант №9**





* 1. Реализовать меню для выбора задачи
  2. Результаты всех вычислений вывести на печать.
  3. Составить систему тестов и выполнить программу для нескольких наборов данных.
  4. Объяснить полученные результаты.

**Анализ первой задачи**

|  |  |  |
| --- | --- | --- |
| Входные данные | Классы входных данных | Выходные данные |
| command – целое число (выбор пункта меню) | 1<= command <=3 | Программа переходит на следующий этап работы |
| command – целое число(выбор пункта меню) | command < 1; command > 3; | Сообщение об ошибке |
| command – целое число(выбор пункта меню) | command = «abc» | Сообщение об ошибке |
| rows – целое число (кол-во строк массива)  columns – целое число (кол-во столбцов массива) | rows <= 0  columns <=0 | Сообщение об ошибке |
| rows – целое число (кол-во строк массива)  columns – целое число (кол-во столбцов массива) | rows > 0  columns > 0 | Создается массив |
| A[i][j] – целое число (элемент массива) | A[i][j] **∈ Z** | Массив успешно заполнен |
| A[i][j] – целое число (элемент массива) | A[i][j] = «abc» | Сообщение об ошибке |

1. Ввести значения переменных, выполнить расчёты и вывести результаты в консоль.
2. В случае ввода некорректных данных, выдать сообщение об ошибке и запросить повторный ввод.

**Анализ второй задачи**

|  |  |  |
| --- | --- | --- |
| Входные данные | Классы входных данных | Выходные данные |
| command – целое число (выбор пункта меню) | 1<= command <=3 | Программа переходит на следующий этап работы |
| command – целое число(выбор пункта меню) | command < 1; command > 3; | Сообщение об ошибке |
| command – целое число(выбор пункта меню) | command = «abc» | Сообщение об ошибке |
| mainString – строка | Числа, буквы латинского и русского алфавита, знаки препинания | Сообщение об успешном заполнении строки |

1. Ввести значение строки, выполнить операции и вывести результаты в консоль.
2. В случае ввода некорректных данных, выдать сообщение об ошибке и запросить повторный ввод.

**Код лабораторной работы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

class Program

{

static void Main(string[] args)

{

int[][] raggedArray = null;

bool shouldClose = false;

string mainString = null;

while (!shouldClose)

{

ShowMenu();

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

Console.Clear();

RaggedArrayWorker.StartWorkRaggedTwoDimArray(ref raggedArray);

Console.Clear();

break;

case 2:

Console.Clear();

StringWorker.StartWork(ref mainString);

Console.Clear();

break;

case 3:

shouldClose = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

}

}

static void ShowMenu()

{

Console.WriteLine("========= Главное меню =========");

Console.WriteLine("1. Работа с рваным массивом");

Console.WriteLine("2. Работа со строками");

Console.WriteLine("3. Выход");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

class RaggedComparer : IComparer<int[]>

{

public int Compare(int[] x, int[] y)

{

if (x.Length > y.Length) return 1;

if (x.Length < y.Length) return -1;

else return 0;

}

}

class RaggedArrayWorker

{

static bool isInitialized = false;

public static void StartWorkRaggedTwoDimArray(ref int[][] raggedTwoDimArray)

{

bool shouldExitInMainMenu = false;

while (!shouldExitInMainMenu)

{

ShowMenu();

PrintStat(ref raggedTwoDimArray);

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

Console.Clear();

CreateArray(ref raggedTwoDimArray);

break;

case 2:

Console.Clear();

ShowFillArrayMenu(ref raggedTwoDimArray);

break;

case 3:

Console.Clear();

PrintArray(ref raggedTwoDimArray);

break;

case 4:

Console.Clear();

RaggedSort(ref raggedTwoDimArray);

break;

case 5:

shouldExitInMainMenu = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

PrintParseErrorMes();

}

}

}

static void PrintParseErrorMes()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

static void PrintStat(ref int[][] raggedTwoDimArray)

{

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Массив[null][null]");

Console.ResetColor();

return;

}

else

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write($"Массив[{raggedTwoDimArray.GetLength(0)}]");

Console.ResetColor();

}

if (isInitialized)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("(Заполнен)");

Console.ResetColor();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("(Пустой)");

Console.ResetColor();

}

}

static void ShowMenu()

{

Console.WriteLine("==== Рваный двумерный массив ====");

Console.WriteLine("1. Создать массив");

Console.WriteLine("2. Заполнить массив");

Console.WriteLine("3. Напечатать массив");

Console.WriteLine("4. Сортировать массив");

Console.WriteLine("5. Выйти в главное меню");

}

static void CreateArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Создание рваного двумерного массива ====\n");

if (raggedTwoDimArray != null)

{

Console.Write("Массив уже создан. Удалить старый и создать новый? (+/-) ");

if (Console.ReadLine() != "+")

{

Console.Clear();

return;

}

}

GetCorrectVal("Введите количество строк(1 - 22360): ", "ERROR: неверное количество строк", out int rows, 1, 22360);

try

{

raggedTwoDimArray = new int[rows][];

Console.WriteLine("1. Строки одинаковой длины");

Console.WriteLine("2. Строки случайной длины(1 - 10)");

Console.WriteLine("3. Задать длину вручную для каждой строки");

GetCorrectVal(">", "Неверный код", out int command, 1, 3);

int columns;

switch (command)

{

case 1:

GetCorrectVal("Введите длину строк(1 - 22360): ", "ERROR: Неверная длина строк", out columns, 1, 22360);

Console.WriteLine("Создание строк...");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

raggedTwoDimArray[i] = new int[columns];

}

break;

case 2:

Console.WriteLine("Создание строк...");

Random rnd = new Random();

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

raggedTwoDimArray[i] = new int[rnd.Next(1, 10)];

}

break;

case 3:

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

GetCorrectVal($"Введите длину строки({i + 1}/{raggedTwoDimArray.GetLength(0)}): ", "ERROR: Неверная длина строки", out columns, 1);

raggedTwoDimArray[i] = new int[columns];

}

break;

}

}

catch (OutOfMemoryException)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("FATAL ERROR: OUT OF MEMORY EXCEPTION!!!\n");

Console.ResetColor();

raggedTwoDimArray = null;

isInitialized = false;

GC.Collect();

return;

}

isInitialized = false;

Console.Clear();

}

private static void ShowFillArrayMenu(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Заполнение двумерного массива ====");

bool shouldExit = false;

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

while (!shouldExit)

{

Console.Write("1. Задать элементы вручную");

int size = 0;

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

{

size += raggedTwoDimArray[i].Length;

}

if (size > 10)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($" ({size} элементов!)");

Console.ResetColor();

}

else Console.WriteLine();

Console.WriteLine("2. Задать элементы через ДСЧ");

Console.WriteLine("3. Назад");

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

ManuallyFillingArray(ref raggedTwoDimArray);

shouldExit = true;

break;

case 2:

RndFillArray(ref raggedTwoDimArray);

shouldExit = true;

isInitialized = true;

break;

case 3:

shouldExit = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else PrintParseErrorMes();

}

Console.Clear();

}

private static void GetCorrectVal(string mess, string errorMes, out int val, int min = int.MinValue, int max = int.MaxValue)

{

bool successInput = false;

val = 0;

while (!successInput)

{

Console.Write(mess);

if (int.TryParse(Console.ReadLine(), out val))

{

if (min <= val && val <= max)

{

successInput = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(errorMes + "\n");

Console.ResetColor();

}

}

else

{

PrintParseErrorMes();

}

}

}

private static void ManuallyFillingArray(ref int[][] raggedTwoDimArray, int rows = 0)

{

Console.WriteLine("\nn - Принудительное завершение ввода");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

bool correctVal = false;

while (!correctVal)

{

Console.Write($"[{i + 1}/{raggedTwoDimArray.GetLength(0)}, {j + 1}/{raggedTwoDimArray[i].Length}] ");

string buf = Console.ReadLine();

if (buf == "n")

{

Console.Clear();

return;

}

if (int.TryParse(buf, out raggedTwoDimArray[i][j]))

{

correctVal = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число");

Console.ResetColor();

}

}

}

}

isInitialized = true;

}

private static void RndFillArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("Массив заполняется...");

Random rnd = new Random();

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

raggedTwoDimArray[i][j] = rnd.Next(1, 100);

}

}

}

static void PrintArray(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Вывод элементов массива ====");

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (!isInitialized)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив пуст!!!\n");

Console.ResetColor();

return;

}

if (raggedTwoDimArray.Length > 1000000)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Массив слишком велик: {raggedTwoDimArray.Length} элементов");

Console.ResetColor();

Console.Write("Вывести массив? (+/-) ");

if (Console.ReadLine() != "+")

{

return;

}

}

Console.Write("Элементы массива:\n");

for (int i = 0; i < raggedTwoDimArray.GetLength(0); i++)

{

for (int j = 0; j < raggedTwoDimArray[i].Length; j++)

{

Console.Write($"{raggedTwoDimArray[i][j]}\t");

}

Console.WriteLine();

}

Console.WriteLine("\n");

}

static void RaggedSort(ref int[][] raggedTwoDimArray)

{

Console.WriteLine("==== Сортировка массива ====");

if (raggedTwoDimArray == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив не создан!!!\n");

Console.ResetColor();

return;

}

if (!isInitialized)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Массив пуст!!!\n");

Console.ResetColor();

return;

}

Console.WriteLine("Массив сортируется...");

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

{

Array.Sort(raggedTwoDimArray[i]);

}

Array.Sort(raggedTwoDimArray, new RaggedComparer());

Console.Clear();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

class StringWorker

{

static string[] sentences =

{

"Если ширина данного поля не указана, то она определяется автоматически - минимально достаточной для изображения значения.",

"Если ширина поля указана и превышает длину помещаемого в поле значения, то при положительной длине поля W значение выравнивается по правой границе.",

"раз два т,ри",

"раз два три.",

"раз два, ,три.",

"раз два три!!!"

};

public static void StartWork(ref string mainString)

{

bool shouldExitInMainMenu = false;

while (!shouldExitInMainMenu)

{

ShowMenu();

Console.Write(">");

if (int.TryParse(Console.ReadLine(), out int command))

{

switch (command)

{

case 1:

Console.Clear();

EnterString(ref mainString);

Console.Clear();

break;

case 2:

Console.Clear();

ReverseWords(ref mainString);

break;

case 3:

Console.Clear();

Console.WriteLine($"Текущая строка: {mainString}");

break;

case 4:

shouldExitInMainMenu = true;

break;

default:

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Неверный код команды\n");

Console.ResetColor();

break;

}

}

else

{

PrintParseErrorMes();

}

}

}

static void PrintParseErrorMes()

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("ERROR: Введённую строку невозможно интерпретировать как целое число\n");

Console.ResetColor();

}

static void PrintErrorMes(string mes)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(mes);

Console.ResetColor();

}

static void ShowMenu()

{

Console.WriteLine("==== Работа со строками ====");

Console.WriteLine("1. Ввести строку");

Console.WriteLine("2. Обработать строку");

Console.WriteLine("3. Напечатать строку");

Console.WriteLine("4. Выйти в главное меню");

}

static void EnterString(ref string mainString)

{

Console.WriteLine("==== Ввод строки ====");

Console.WriteLine("1. Ввести вручную");

Console.WriteLine("2. Выбрать из массива");

GetCorrectVal(out int command, ">", "Неверный код команды", 1, 2);

switch (command)

{

case 1:

Console.Write("Введите строку: ");

string buf = Console.ReadLine();

if (buf.Length > 0) mainString = buf;

else

{

mainString = null;

PrintErrorMes("ERROR: Введена пустая строка");

}

break;

case 2:

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

{

Console.WriteLine($"{i + 1}. {sentences[i]}");

}

GetCorrectVal(out int sentNum, "Введите номер предложения: ", "Неверный номер предложения", 1, sentences.Length);

mainString = sentences[sentNum - 1];

break;

}

}

private static void GetCorrectVal(out int val, string mess, string errorMes, int min = int.MinValue, int max = int.MaxValue)

{

bool successInput = false;

val = 0;

while (!successInput)

{

Console.Write(mess);

if (int.TryParse(Console.ReadLine(), out val))

{

if (min <= val && val <= max)

{

successInput = true;

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(errorMes + "\n");

Console.ResetColor();

}

}

else

{

PrintParseErrorMes();

}

}

}

static int GetNumMarks(ref string word)

{

int numMarks = 0;

int i = word.Length - 1;

char[] punctMarks = { ',', '.', '!', '?' };

while ((Array.IndexOf(punctMarks, word[i]) > -1) && i > 0)

{

numMarks++;

i--;

}

for (i--; i >= 0; i--)

{

if (Array.IndexOf(punctMarks, word[i]) != -1)

{

return -1;

}

}

return numMarks;

}

static void ReverseWords(ref string mainString)

{

if (mainString == null)

{

PrintErrorMes("ERROR: Пустая строка!");

return;

}

char[] delimiterChars = {' '};

string[] words = mainString.Split(delimiterChars);

int wordNumber = 0;

List<string> incorrectWords = new List<string>(words.Length);

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

{

if (words[i] != "")

{

wordNumber++;

int numMarks = GetNumMarks(ref words[i]);

if (numMarks > -1 && words[i].Length - numMarks == wordNumber)

{

int bufLength = words[i].Length - numMarks;

char[] buf = new char[words[i].Length];

//Reverse word in buffer array

for (int j = 0; j < bufLength; j++)

{

buf[j] = words[i][bufLength - j - 1];

}

//Add punctuation marks

for (int j = bufLength; j < buf.Length; j++)

{

buf[j] = words[i][j];

}

words[i] = new string(buf);

}

if (numMarks == -1) incorrectWords.Add(words[i]);

}

}

mainString = String.Join(" ", words);

Console.WriteLine($"Предложение: {mainString}");

if (incorrectWords.Count > 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write($"Слова с опечатками: ");

Console.ResetColor();

foreach (var word in incorrectWords)

{

Console.Write($"({word}); ");

}

Console.WriteLine("\n");

}

}

}

}

**Блок – схемы функций**

