Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра « »

Лабораторная работа №2

по дисциплине «Теория информации»

«RLE»

Выполнил: студент гр. −31 Д. Ю.

Проверил: Н. В.

Тамбов,

***Цели и задачи.***

Разработать приложение, реализующее алгоритм кодирования серий.

***Теория.***

Алгоритм RLE является, наверное, самым простейшим из всех: суть его заключается в кодировании повторов. Другими словами, мы берём последовательности одинаковых элементов, и «схлопываем» их в пары «количество/значение». Например, строка вида «AAAAAAAABCCCC» может быть преобразована в запись вроде «8×A, B, 4×C». Это, в общем-то, всё, что надо знать об алгоритме.

***Решение задачи****.*

В программе рисуем аватарку, сохраняем в файл 1.txt. Здесь же сжимаем изображение и получаем файл 2.txt. Обратный процесс по восстановлению исходного изображения начинаем с загрузки сжатого файла 2.txt, после чего выполняется декодирование с результатом, записанным в файл 3.txt.

Метод, отвечающий за сжатие исходного изображения:

/// <summary>

/// Выполнить компрессию содержимого массива array

/// Схема 1

/// </summary>

/// <param name="array">Что сжимать</param>

/// <param name="x">Размер массива на входе</param>

/// <param name="y">Размер массива на входе</param>

/// <param name="razd1">первый разделитель</param>

/// <param name="razd2">второй разделитель</param>

/// <returns>Вернет сжатый массив</returns>

**internal** int**[,]** Compress**(**int**[,]** array**,** int x**,** int y**,** char razd1**,** char razd2**)**

**{**

//Разделители: \n=n // \t=t // \r=r

var razdel1 **=** RChar**(**razd1**);**

var razdel2 **=** RChar**(**razd2**);**

int pls **=** razdel1**!=**""**?** 3 **:** 2**;**

**if** **(**razdel2 **!=** ""**)** pls**++;**

//Выходной массив может быть Меньше или Больше Входного

int maxXLength **=** 0**;**//самая длинная строка

int povtorVal **=** 0**;**//счетчик повторов

int tmpVal **=** 0**;**//временный символ

List**<**string**>** outarr **=** **new** List**<**string**>();**

List**<**int**>** val **=** **new** List**<**int**>();**

int**[]** len **=** **new** int**[**y**];**

int**[,]** retArr **=** **new** int**[**x **\*** pls**,** y**];**

//у строк по х столбцов

**for** **(**int i **=** 0**;** i **<** y**;** i**++)**

**{**

**for** **(**int j **=** 0**;** j **<** x**;** j**++)**

**{**

**if** **(**j **==** 0**)**

**{**

tmpVal **=** array**[**j**,** i**];**//начало строки берем первый символ

len**[**i**]** **=** maxXLength**;**

povtorVal **=** 0**;**

povtorVal**++;**

Koeff **+=** maxXLength**;**

maxXLength **=** 0**;**

**}**

**else**//иначе сравним символ с тикущим

**{**

**if** **(**tmpVal **==** array**[**j**,** i**])**//если предыдущий равен текущему

**{**

povtorVal**++;**

**if** **(**povtorVal **==** x**)**//если вся строка из повторного символа

**{**

//outarr.Add(povtorVal + razdel1 + tmpVal + razdel2);

val**.**Add**(**povtorVal**);**

**if** **(**razdel1 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel1**));**

val**.**Add**(**tmpVal**);**

**if** **(**razdel2 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel2**));**

povtorVal **=** 0**;**

maxXLength **+=** pls**;**

len**[**i**]** **=** maxXLength**;**

**}**

**else** **if** **(**j **==** x **-** 1 **&&** povtorVal **>** 0**)**//не вся строка но конец

**{**

//outarr.Add(povtorVal + razdel1 + tmpVal + razdel2);

val**.**Add**(**povtorVal**);**

**if** **(**razdel1 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel1**));**

val**.**Add**(**tmpVal**);**

**if** **(**razdel2 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel2**));**

povtorVal **=** 0**;**

maxXLength **+=** pls**;**

len**[**i**]** **=** maxXLength**;**

**}**

**}**

**else**//неравен

**{**

**if** **(**povtorVal **==** 0**)** povtorVal**++;**

//outarr.Add(povtorVal + razdel1 + tmpVal + razdel2);

val**.**Add**(**povtorVal**);**

**if(**razdel1**!=**""**)** val**.**Add**(**Convert**.**ToChar**(**razdel1**));**

val**.**Add**(**tmpVal**);**

**if** **(**razdel2 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel2**));**

maxXLength **+=** pls**;**

len**[**i**]** **=** maxXLength**;**

tmpVal **=** array**[**j**,** i**];**

**if** **(**j **==** x **-** 1**)**

**{**

//outarr.Add(1 + razdel1 + tmpVal + razdel2);

val**.**Add**(**1**);**

**if** **(**razdel1 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel1**));**

val**.**Add**(**tmpVal**);**

**if** **(**razdel2 **!=** ""**)** val**.**Add**(**Convert**.**ToChar**(**razdel2**));**

maxXLength **+=** pls**;**

len**[**i**]** **=** maxXLength**;**

**}**

povtorVal **=** 1**;**

**}**

**}**

**}**

int er **=** 0**;**

//переход на след строку//заполнить массив сжатыми значениями

**foreach** **(**int i1 **in** val**)**

**{**

retArr**[**er**,** i**]** **=** i1**;**

er**++;**

**}**

val**.**Clear**();**

**}**

maxXLength **=** len**.**Max**();**

XY **=** **new** Point**(**y**,** x **\*** pls**);**

ArrToSaveFile **=** retArr**;**

**if** **(**x **\*** y **\*** pls**>=** Koeff**)**

**{**

double tmp **=** **(**Koeff **/** **(**x **\*** y **\*** pls**))** **\*** 100.0**;**

Koeff **=** Math**.**Round**(**tmp**,** 2**);**

**}**

**else**

**{**

double tmp **=** **((**x **\*** y **\*** pls**)** **/** Koeff**)** **\*** 100.0**;**

Koeff **=** Math**.**Round**(**tmp**,** 2**);**

**}**

len **=** **null;**

outarr**.**Clear**();**

**return** retArr**;**

**}**

Метод, выполняющий восстановление сжатой последовательности:

/// <summary>

/// Восстановить сжатые последовательности

/// На вход массив из clFile->LoadProjectCoder()

/// </summary>

**internal** int**[,]** DeCodArr**(out** int xLen**,** **out** int yLen**,** int**[,]** codArr**,** Point xyCodArr**,** int sxemaRazbora**)**

**{**

//sxemaRazbora влияет на то в какой ячейке\ках codArr ставится разделитель

//sxemaRazbora=2::[povt|0][znak|1][razd2|2]

//значения в sxemaRazbora=2 не распаковывать\не обрабатывать на вывод

//№-строки И int[] значений

Dictionary**<**int**,** int**[]>** decod **=** **new** Dictionary**<**int**,** int**[]>();**

int maxStrLen **=** 0**;**//самая длинная цепоцка

**for** **(**int j **=** 0**;** j **<** xyCodArr**.**Y**;** j**++)**//по строке

**{**

string posled **=** ""**;**

**for** **(**int i **=** 0**;** i **<** xyCodArr**.**X**;** i **+=** **(**sxemaRazbora **+** 1**))** //по ячейке в строке

**{**

//повторы

int colPovtr **=** codArr**[**i**,** j**];** //0//3

//знак

int znak **=** codArr**[**i **+** 1 **,** j**];** //0+1//3+1

//распаковать знак N-раз

**for** **(**int k **=** 0**;** k **<** colPovtr**;** k**++)**

**{**

posled **+=** znak**;**//цепочка из 0 и 1

**}**

**}**

**if** **(**maxStrLen **<** posled**.**Length**)** maxStrLen **=** posled**.**Length**;**

var tmp **=** **new** int**[**posled**.**Length**];**

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

**{**//пишем по символам//в строке 0 или 1

tmp**[**i**]** **=** int**.**Parse**(**posled**[**i**].**ToString**());**

**}**

decod**.**Add**(**j**,** tmp**);**

**}**

int**[,]** array **=** **new** int**[**maxStrLen**,** xyCodArr**.**Y**];**

//перегнать <j, int[i]> в int[i, j]

**for** **(**int j **=** 0**;** j **<** xyCodArr**.**Y**;** j**++)**

**{**

int**[]** tmp **=** decod**[**j**];** //j-строка = []

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

**{**

array**[**i**,** j**]** **=** tmp**[**i**];**

**}**

**}**

//выходные значения для холста

xLen **=** maxStrLen**;**

yLen **=** xyCodArr**.**Y**;**//строк

decod**.**Clear**();**

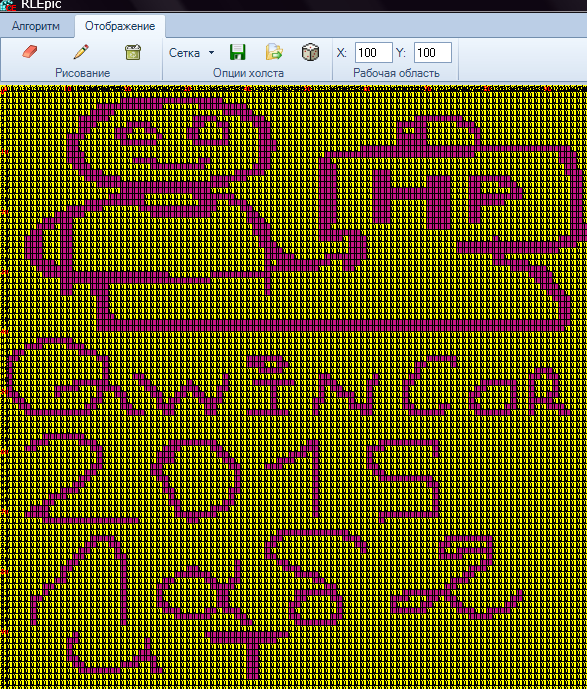
//на холст

**return** array**;**

**}**

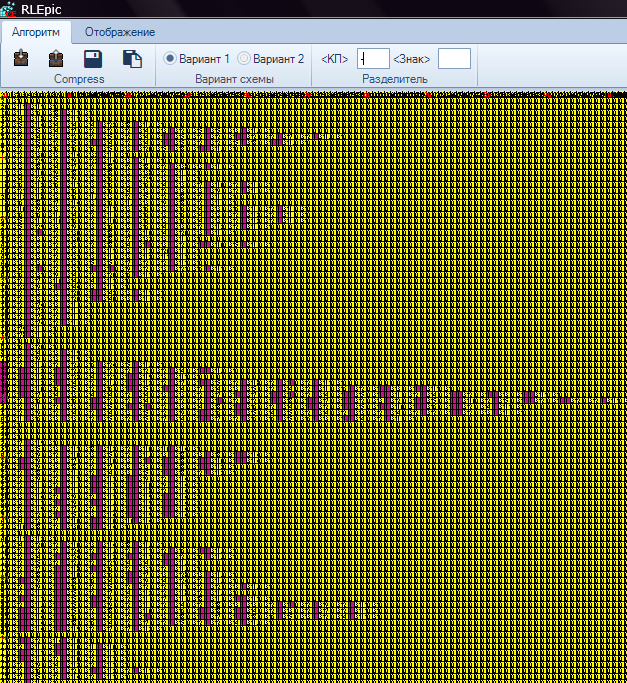
***Выбор языка программирования***

Язык C#, самый легкий и больше всего изученный мною.

***Тестирование программы***

В несжатом виде файл имеер размер 9,96 КБ.

После обработки алгоритмом rle файл уменьшился до 4,53 КБ.

По «всплескам» видно, что сжатие на разных участках происходит по-разному и это логично, так как алгоритм хорош на объемных участках с монотонной информацией, тогда как чередование, например линии, вызывает не сжатие, а разрастание выходной последовательности.