**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Ухтинский государственный технический университет»**

**(УГТУ)**

Кафедра вычислительной техники, информационных систем и технологий

**КОНТРОЛЬНАЯ РАБОТА**

Дисциплина: «Теория информации»

Шифр: 191407 Группа ИСТ-2-19 Курс 2

Морданов Егор Владимирович

Проверил:

доцент кафедры ВТИСиТ Куделин А. Г.

Ухта

2020

содержание

[Введение 3](#_Toc61543363)

[1 Алгоритм LZ-78 4](#_Toc61543364)

[2 Программа кодирования алгоритмом LZ-78 6](#_Toc61543365)

[2.1 Реализация алгоритма LZ78 6](#_Toc61543366)

[2.2 Ввод исходного текста из файла 7](#_Toc61543367)

[2.3 Прочие функции 9](#_Toc61543368)

[Заключение 12](#_Toc61543369)

[Список использованной литературы 13](#_Toc61543370)

[Приложения 14](#_Toc61543371)

[Приложение 1 Листинг программы 14](#_Toc61543372)

# Введение

Целью данной контрольной работы является написание программы, производящей кодирование текста при помощи алгоритма LZ78, в среде разработки Embarcadero RAD STUDIO на языке программирования C++.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Изучить принцип работы алгоритма LZ78, его особенности и отличия от других алгоритмов;
* Реализовать алгоритм LZ78 в среде разработки Embarcadero RAD STUDIO на языке программирования C++, а именно – кодирование текста с возможностью его последующего декодирования;
* Реализовать возможность ввода текста из файла;
* Реализовать дополнительные функции, такие как: возможность задать размер словаря, пошаговая работа алгоритма, просмотр словаря, справка, вывод коэффициента сжатия.

# Алгоритм LZ-78

LZ78 — алгоритм сжатия без потерь, опубликованный в статьях израильских математиков Авраама Лемпеля, Яакова Зива и Мэтта Махони в 1978 году. Алгоритмы LZ78 и LZ77 — наиболее известные варианты в семействе LZ\*, которое включает в себя также LZW, LZSS, LZMA и другие алгоритмы.

Главное отличие метода LZ78 от LZ77 в том, что LZ78 не использует буфер поиск, упреждающий буфер и скользящее окно.

Вместо этого в методе LZ78 имеется словарь встретившихся ранее строк. В начале этот словарь пуст (или почти пуст), и размер этого словаря ограничен только объемом доступной памяти. На выход кодера поступает последовательность меток, состоящих из двух полей. Первое поле — это указатель на строку в словаре, а второе код символа. Метка не содержит длины строки, поскольку строка берется из словаря. Каждая метка соответствует последовательности во входном файле, и эта последовательность добавляется в словарь после того, как метка записана в выходной сжатый файл. Ничего из словаря не удаляется, что является одновременно и преимуществом над LZ77 (поскольку будущие строки могут совпадать даже с очень давними последовательностями) и недостатком (так как быстро растет объем словаря).

Словарь начинает строиться из пустой строки в позиции нуль. По мере поступления и кодирования символов, новые строки добавляются в позиции 1, 2 и т.д. Когда следующий символ  читается из входного файла, в словаре ищется строка из одного символа . Если такой строки нет, то  добавляется в словарь, а на выход подается метка . Эта метка означает строку «нуль » (соединение нулевой строки и ). Если вхождение символа  обнаружено (скажем, в позиции 37), то читается следующий символ , и в словаре ищется вхождение двухсимвольной строки . Если такое не найдено, то в словарь записывается строка , а на выход подается метка . Такая метка означает строку , так как позицию 37 в словаре занимает символ . Процесс продолжается до конца входного файла.

В общем случае текущий символ читается и становится однобуквенной строкой. Затем кодер пытается найти ее в словаре. Если строка найдена, читается следующий символ и присоединяется к текущей строке, образуя двухбуквенную строку, которую кодер опять пытается найти в словаре. До тех пор, пока такие строки находятся в словаре, происходит чтение новых символов и их присоединение к текущей строке. В некоторый момент такой строки в словаре не оказывается. Тогда кодер добавляет ее в словарь и строит метку, в первом поле которой стоит указатель на последнюю найденную в словаре строку, а во втором поле записан последний символ строки (на котором произошел обрыв успешных поисков).

На каждом шаге строка, добавленная в словарь, совпадает с кодируемой строкой минус последний символ. В типичном процессе сжатия словарь начинается с коротких строк, но по мере продвижения по кодируемому тексту, все более и более длинные строки добавляются в словарь. Размер словаря может быть фиксированным или определяться размером доступной памяти каждый раз, когда запускается программа сжатия LZ78. Большой словарь позволяет делать глубокий поиск длинных совпадений, но ценой этого служит длина поля указателей (а, значит, и длина метки) и замедление процесса словарного поиска.

# Программа кодирования алгоритмом LZ-78

## Реализация алгоритма LZ78

Непосредственно процесс кодирования производит функция vector<node> lz78(wstring s, int sz = 0){…}.

Для реализации алгоритма используются дополнительные библиотеки <vector> и <map>, которые добавляют удобные в использовании структуры данных. Vector – динамический массив с готовыми функциями, он используется для хранения закодированной информации. Map – еще одна вариация массива с готовыми функциями, map не имеет ограничений на тип значений ключа, при помощи чего удобно хранить информацию о словаре.

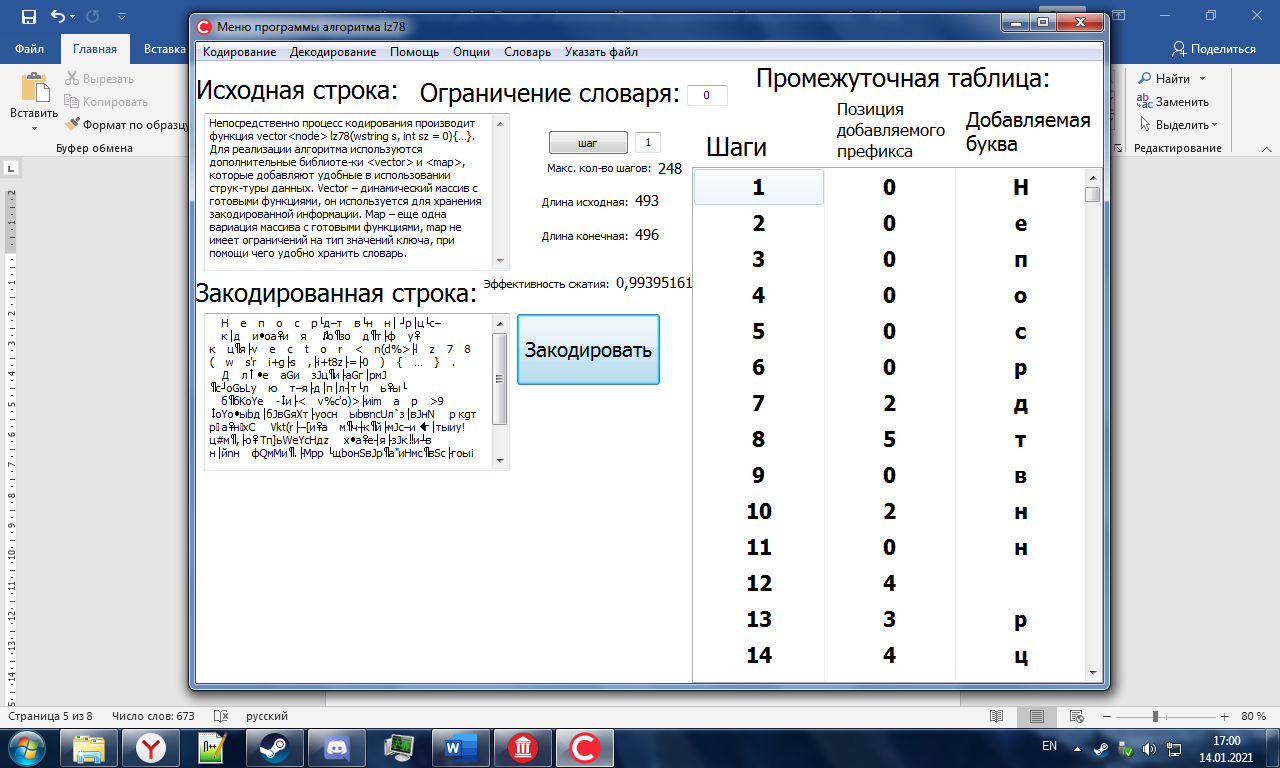


Рисунок 1 - Кодирование текста

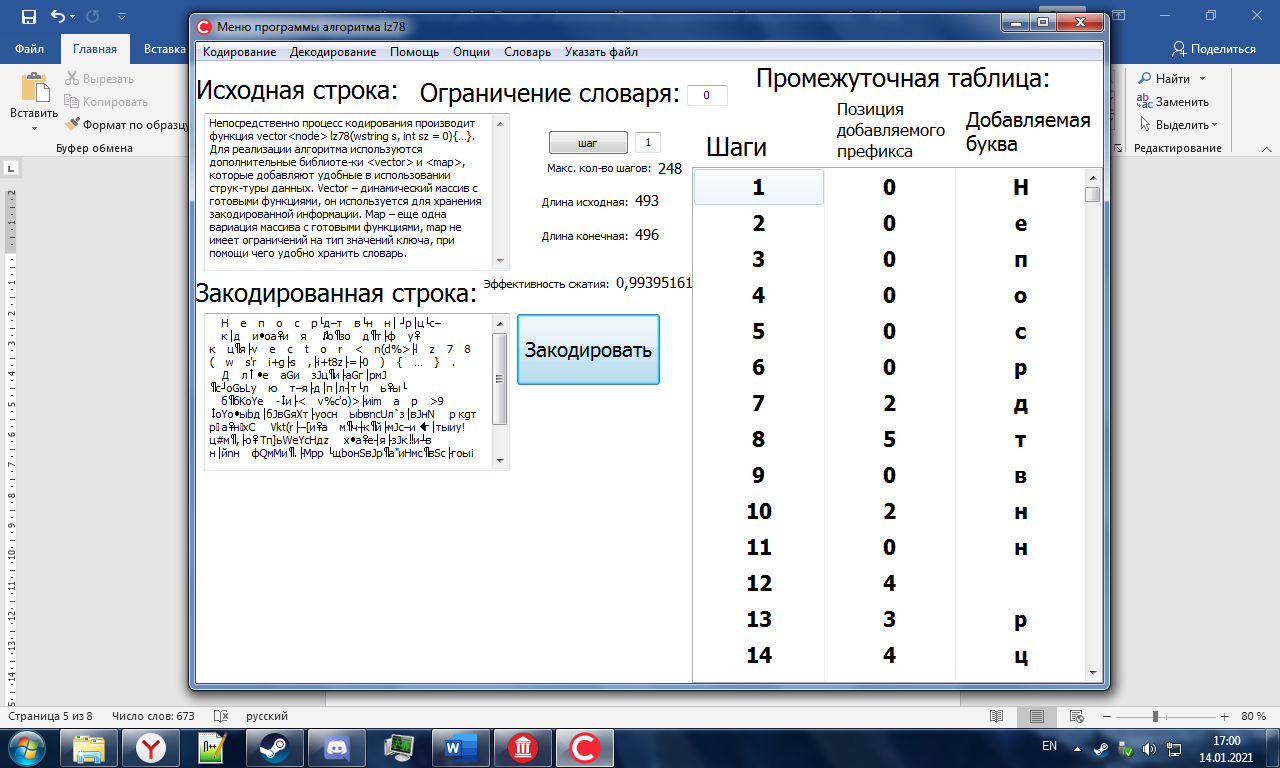


Рисунок 2 - Декодирование текста

## Ввод исходного текста из файла

Ввод исходного текста из файла производит процедура void \_\_fastcall TForm1::N7Click(TObject \*Sender){…}. Она получает имя файла при помощи компонента OpenDialog, после чего построчно считывает и записывает содержимое файла в соответствующее поле.

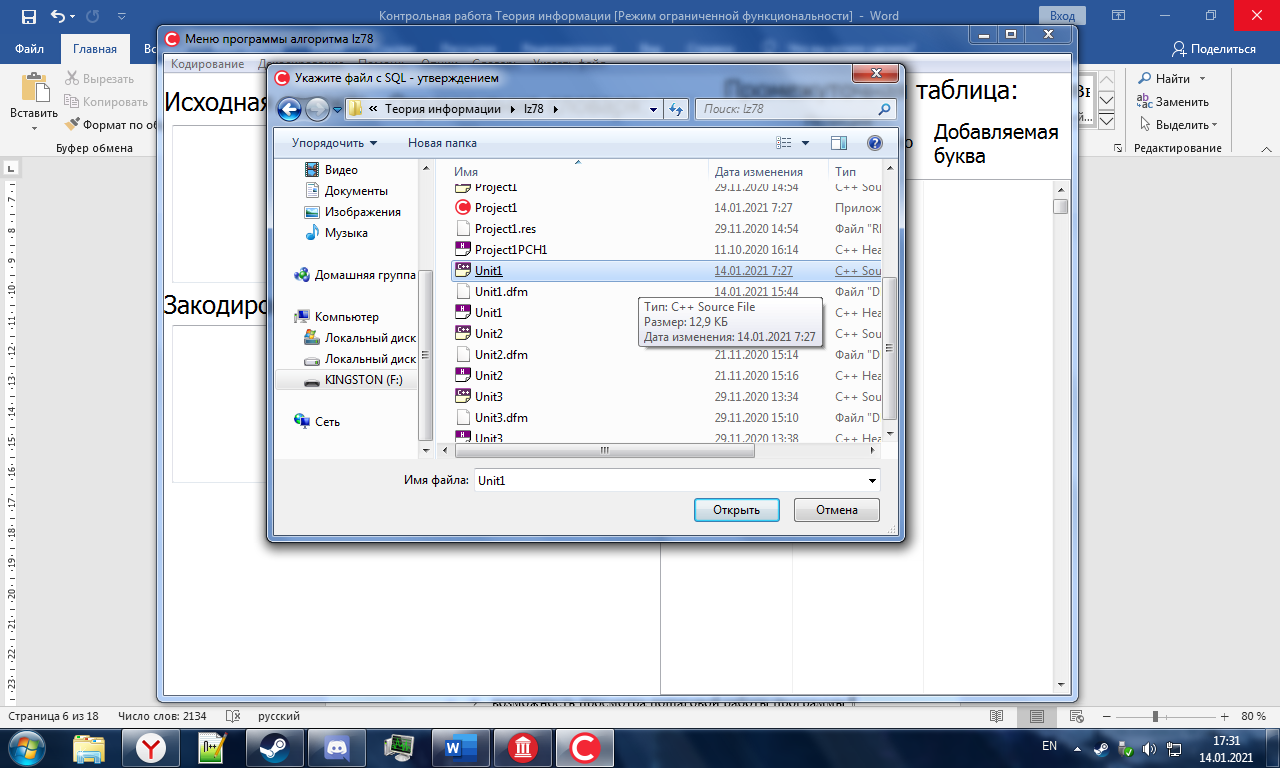


Рисунок 3 - Диалог выбора файла

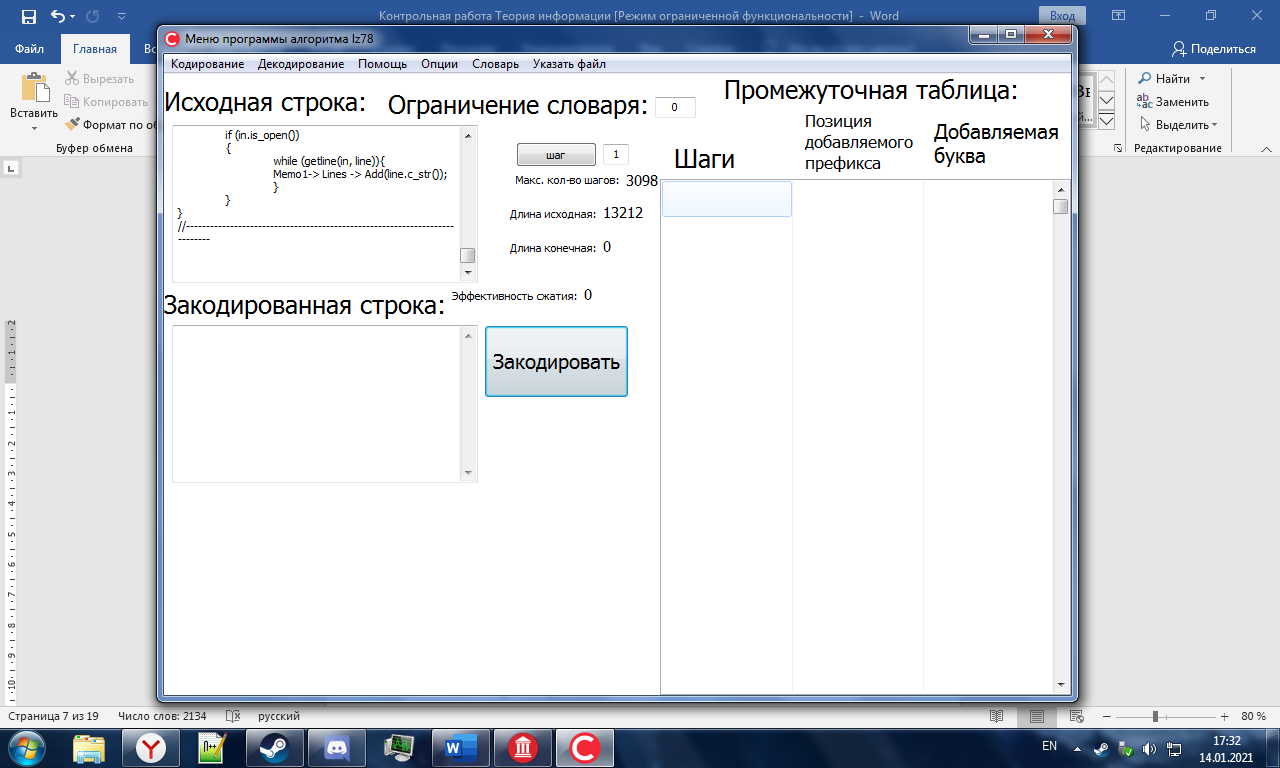


Рисунок 4 - Результат считывания файла

## Прочие функции

Дополнительно в программе реализованы:

* Возможность задать ограничение словаря. По умолчанию установлено значение 0 – без ограничений;
* Возможность просмотра пошаговой работы программы;
* Возможность просмотра словаря;
* Справка – новое окно с информацией выводится процедурой void \_\_fastcall TForm1::N3Click(TObject \*Sender){…};
* Светлая и темная темы, за переключение которых через меню отвечает процедура void \_\_fastcall TForm1::N6Click(TObject \*Sender){…};
* Вывод на экран дополнительной информации о кодировании, такой как длина исходного и закодированного сообщений, количество шагов алгоритма, эффективность сжатия;

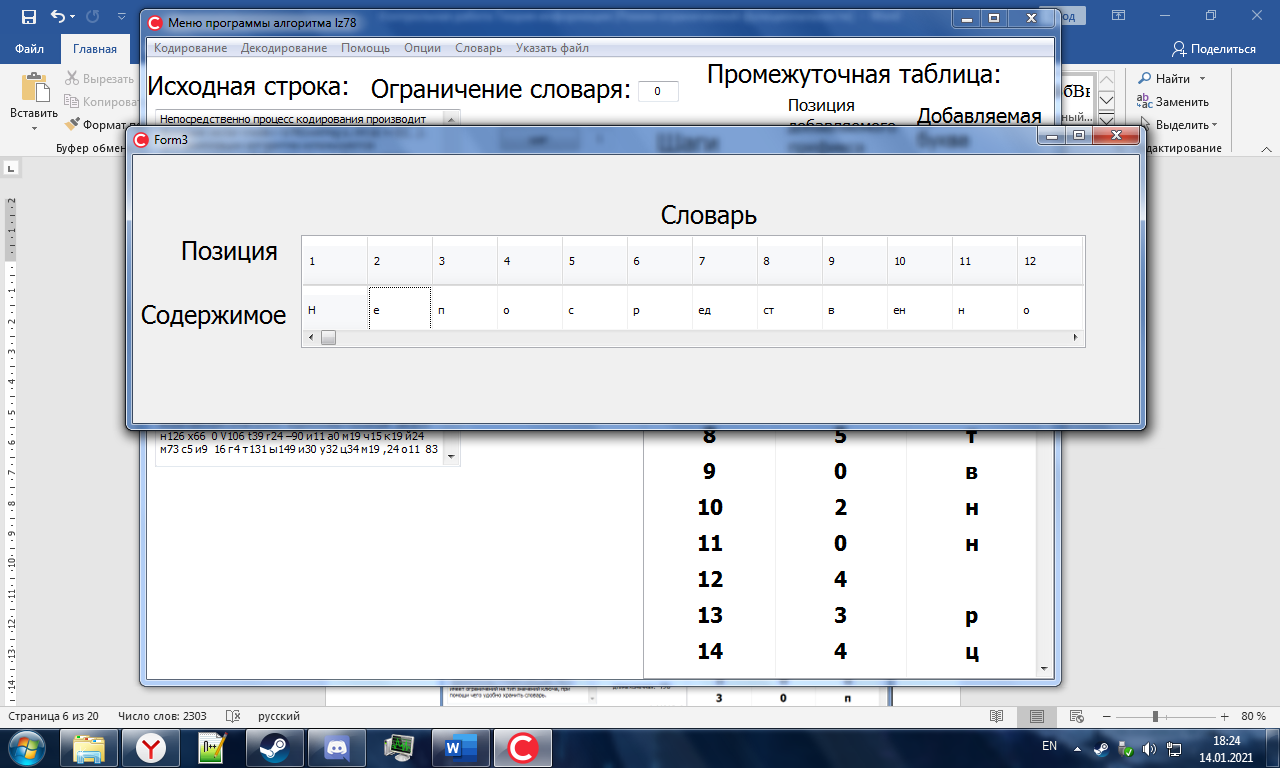


Рисунок 5 - Просмотр словаря

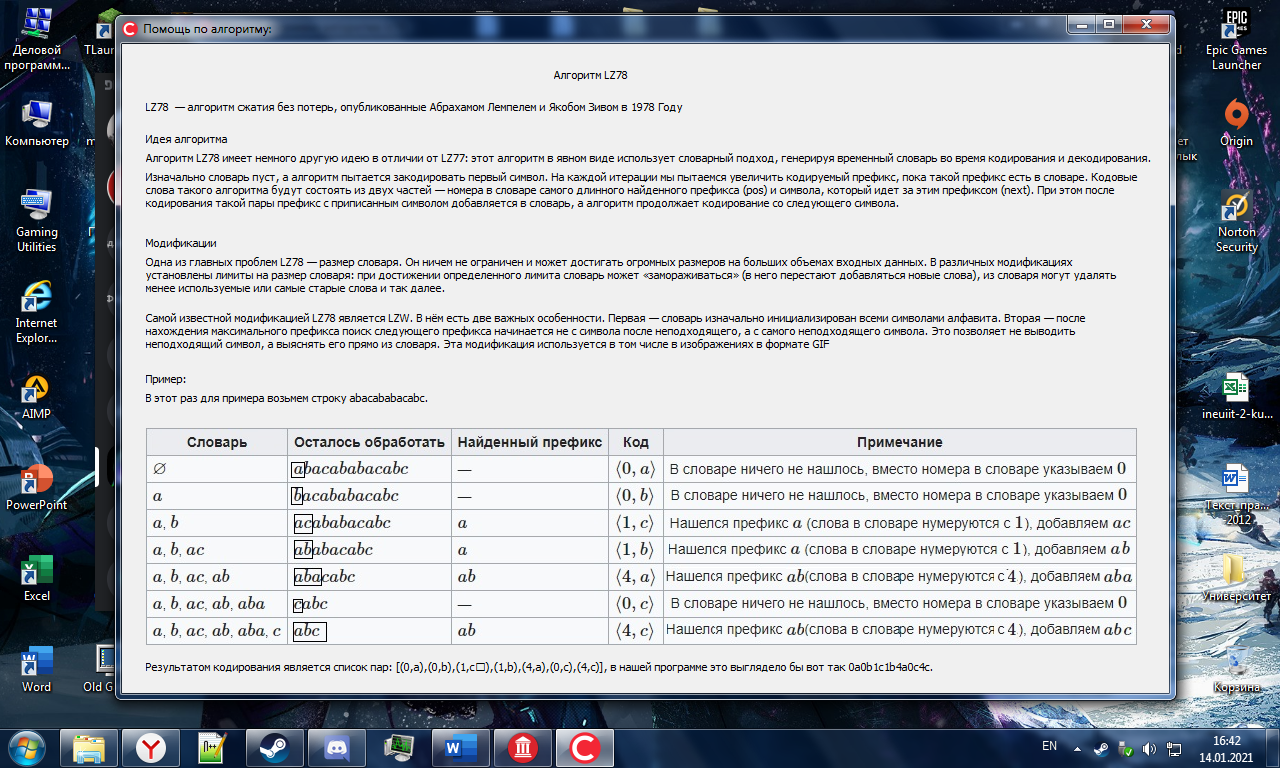


Рисунок 6 - Справка

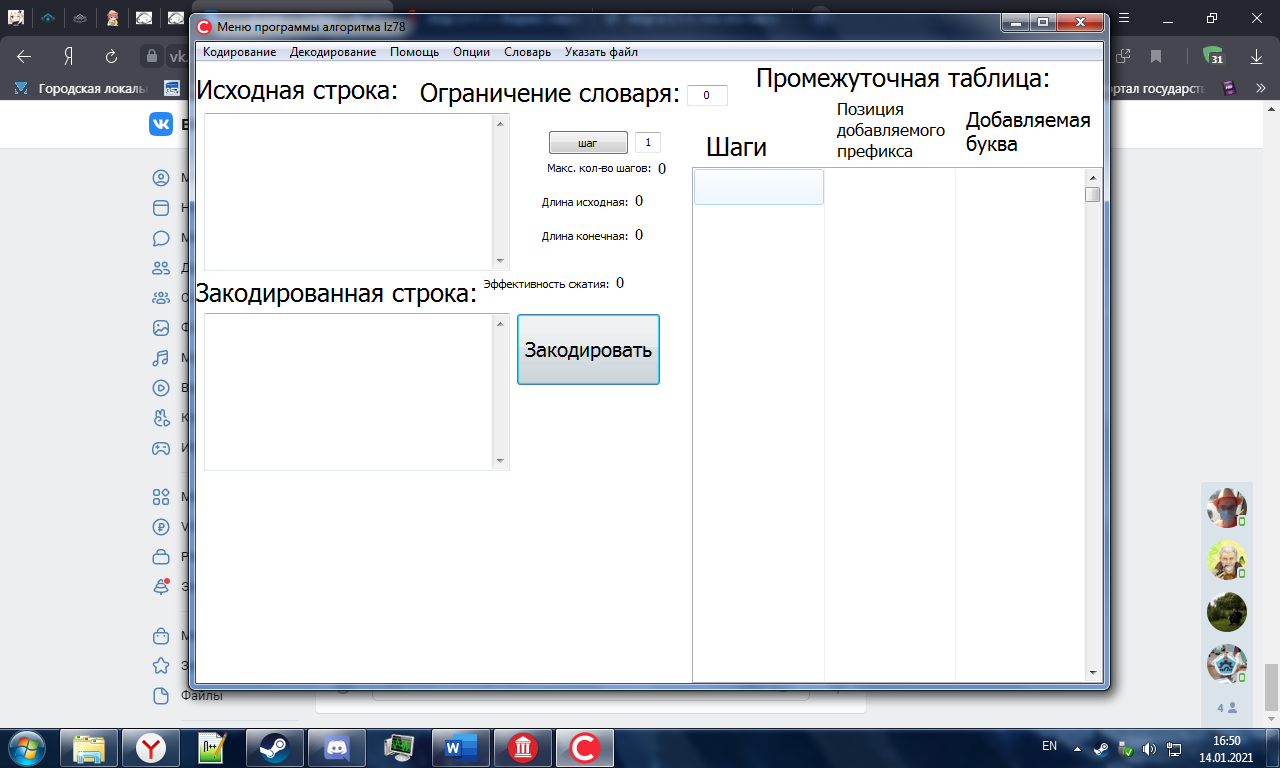


Рисунок 7 - Светлая тема

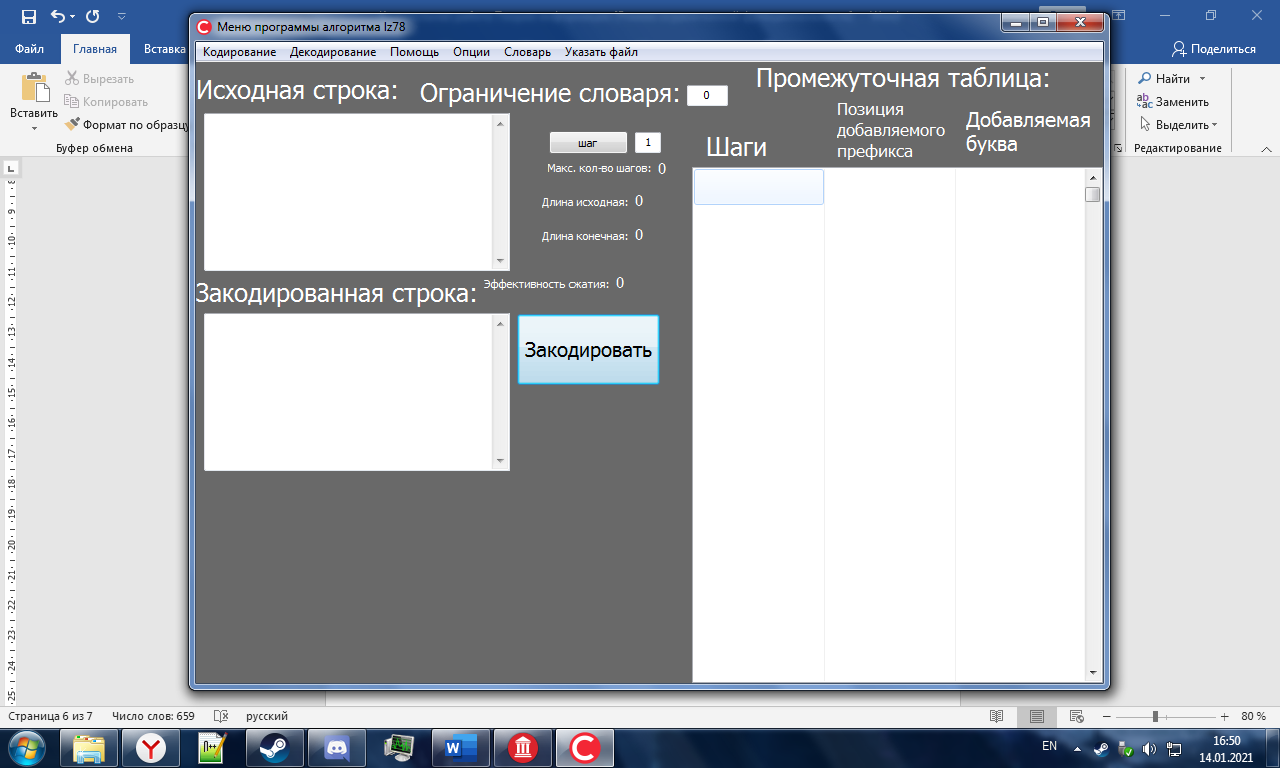


Рисунок 8 - Темная тема

# Заключение

При выполнении контрольной работы были изучен принцип работы алгоритма LZ78, его особенности и отличия от других алгоритмов. Данный алгоритм был реализован в среде разработки Embarcadero RAD STUDIO на языке программирования C++.

Полученная программа кодирует текст при помощи алгоритма LZ78, с возможностью его последующего декодирования и возможностью загружать исходный текст из файла. Также пользователь может задать ограничение словаря для алгоритма, посмотреть словарь, получить информацию об алгоритме при помощи встроенной справки, выбрать темную или светлую тему и увидеть информацию об эффективности кодирования текста алгоритмом. Программа умеет пошагово выводить результаты работы.

Все поставленные задачи были выполнены, а цель достигнута.

# Список использованной литературы

1. LZ77. – Текст : электронный // Википедия : [сайт]. – URL : https://ru.wikipedia.org/. – Дата обращения: 24.10.2020.
2. LZ78. – Текст : электронный // scask.ru : [сайт]. – URL : https://scask.ru/. – Дата обращения: 24.10.2020.
3. Лекция 13. – Текст : электронный // Студопедия : [сайт]. – URL : https://studopedia.su/. – Дата обращения: 24.10.2020.

# Приложения

## Приложение 1 Листинг программы

|  |
| --- |
| //---------------------------------------------------------------------------  #include <vcl.h>  #pragma hdrstop  #include <string>  #include <iostream>  #include <map>  #include <vector>  #include <algorithm>  #include <iostream>  #include <fstream>  #include "Unit1.h"  #include "Unit2.h"  #include "Unit3.h"  using namespace std;  //---------------------------------------------------------------------------  #pragma package(smart\_init)  #pragma resource "\*.dfm"  TForm1 \*Form1;  struct node {  int pos;  wchar\_t next;  };  bool bettercompression(false),darktheme(false),limitdictionary(false);  vector<node> lz78(wstring s, int sz = 0) {  vector<node>ans; // закодированная строка  int i = 0;  while(i < s.size()) {  wstring buffer;  map<wstring, int>m; // используется для создания пар  for (; i < s.size(); ++i) {  if (m.count(buffer + s[i])) // m.count смотрит по первому параметру и возвращает 1  buffer += s[i]; // если эта строка есть и 0 если её нет  else {  ans.push\_back({ m[buffer],s[i] }); // добавляет в ans буффер + след символ  m[buffer + s[i]] = m.size(); // распределяет "номера" в таблице  buffer.clear(); // очищает буффер  if (sz > 0 && sz == m.size()) {  ++i;  break;  }  }  }  if (!buffer.empty()) { // если в буфере остались буквы после того как обработали всю строку |

Продолжение приложения 1

|  |
| --- |
| wchar\_t l = buffer.back(); // берем посл. букву и кладём её в вектор ans  buffer.pop\_back();  ans.push\_back({ m[buffer], l });  }  if (sz > 0 && sz == m.size())  ans.push\_back({7, 7});  }  return ans;  }  wstring decode(vector<node> enc) {  wstring ans; // декодируемая строка  int i = 0;  while(i < enc.size()) {  vector<wstring>v(1); // создает на 0 позиции пустой элемент чтобы не сдох вектор при его самой первой иницилизации  for (; i < enc.size(); i++) {  if (enc[i].pos == 7 && enc[i].next == 7)  break;  wstring word = v[enc[i].pos] + enc[i].next; // обращается по позиции + добавляет символ (поэтому и нужен 0 позиции с пустым элементом чтобы он добавил ""+"a"  ans += word; // декодируемая строка  v.push\_back(word); //создает "словарь" по ходу действия  Form3->StringGrid1 -> Cells[i][0] = v.size()-1;  Form3->StringGrid1 -> Cells[i][1] = v.back().c\_str();  }  ++i;  }  return ans;  }  //------------------------------------------------------------------------------------------  //---------------------------------------------------------------------------  \_\_fastcall TForm1::TForm1(TComponent\* Owner)  : TForm(Owner)  {  }  //---------------------------------------------------------------------------  int numconvert(string s) {  int ans = 0;  for (int i = 0; i < s.size(); ++i) {  if (s[i] >= '0' && s[i] <= '9') {  ans \*= 10;  ans += s[i] - '0';  }  else  return 0;  }  return ans;  }  void \_\_fastcall TForm1::Button1Click(TObject \*Sender)  {  int limit = numconvert(AnsiString(Edit1 -> Text).c\_str())+1;  if (limit > 0)  limitdictionary = true;  else |

Продолжение приложения 1

|  |
| --- |
| limitdictionary = false;  for (int i=0; i < 3; i++) {  StringGrid1 -> Cols[i]->Clear();  StringGrid1 -> Rows[i]->Clear();  }  int counter;  wstring memobuffer;  counter=0;  Memo2 -> Text="";  vector <node> ans = lz78(Memo1 -> Text.c\_str(), limit);  string val;  wstring wval;  for (int i=0;i < ans.size(); i++)  {  StringGrid1 -> Cells[0][counter] = i+1;  StringGrid1 -> Cells[1][counter] = ans[i].pos;  StringGrid1 -> Cells[2][counter] = ans[i].next;  if (bettercompression){  memobuffer+=wchar\_t(ans[i].pos+1);// 0 для строки в c++ считается концом ввода поэтому +1  memobuffer+=ans[i].next; // так же кодируем цифры наподобие 13 в один символ для еще большего сжатия  }  else{  val = to\_string(ans[i].pos);  wval = wstring(val.begin(),val.end());  memobuffer += wval;  memobuffer +=L" ";  memobuffer +=ans[i].next;  }  counter++;  }  Memo2 -> Text=memobuffer.c\_str();  decode(ans);  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button2Click(TObject \*Sender)  {  if (bettercompression) {  wstring s = Memo2 -> Text.c\_str(); // берем закодированную строку из 2 панели  vector<node>a; // создаем массив для декодирования  for (int i = 0; i < s.size(); i += 2) // заполняем массив  a.push\_back({ s[i]-1, s[i + 1] }); // -1 для того чтобы исправить костыль с 0 в строке c++ берем символ позиции и следующим символ строки  Memo3 -> Text =decode(a).c\_str(); // выводим декодированную строку на 3 панель  }  else {  wstring s = Memo2 -> Text.c\_str();  vector<node>a; |

Продолжение приложения 1

|  |
| --- |
| int i = 0;  while (i < s.size()) {  int cv = 0;  while(i < s.size() && s[i] >= L'0' && s[i] <= L'9')  cv \*= 10, cv += s[i] - L'0', ++i;  ++i;  a.push\_back({ cv, s[i] });  ++i;  }  Memo3 -> Text =decode(a).c\_str();  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Debug1Click(TObject \*Sender)  {  Form3->Visible = true;  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::N1Click(TObject \*Sender)  {  Button1->Visible = true;  Button2->Visible = false;  Label1->Visible = true;  Label2->Visible = true;  Label3->Visible = false;  Label4->Visible = true;  Label5->Visible = true;  Label6->Visible = true;  Label7->Visible = true;  Memo1->Visible = true;  Memo2->Visible = true;  Memo3->Visible = false;  StringGrid1->Visible = true;  Button2->Top = 259;  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::N2Click(TObject \*Sender)  {  Button1->Visible = false;  Button2->Visible = true;  Label1->Visible = false;  Label2->Visible = true;  Label3->Visible = true;  Label4->Visible = false;  Label5->Visible = false;  Label6->Visible = false;  Label7->Visible = false;  Memo1->Visible = false;  Memo2->Visible = true;  Memo3->Visible = true;  StringGrid1->Visible = false; |

Продолжение приложения 1

|  |
| --- |
| Button2->Top = 259;  if (Button2->Visible and bettercompression) {  wstring s = Memo2 -> Text.c\_str();  Label10->Caption=s.size()\*2;  }  if (Button2->Visible and !bettercompression) {  wstring s = Memo2 -> Text.c\_str();  int i=0,j = 0;  while (i <s.size()) {  int cv = 0;  while(i < s.size() && s[i] >= L'0' && s[i] <= L'9')  cv \*= 10, cv += s[i] - L'0', ++i;  ++i;  ++i;  j++;  }  Label10->Caption=j;  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::N3Click(TObject \*Sender)  {  Form2->Visible = true;  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::N4Click(TObject \*Sender)  {  if (bettercompression)  {bettercompression = false;  N4->Caption="Более лучшее сжатие(Вкл)";}  else{  bettercompression = true;  N4->Caption="Более лучшее сжатие(Выкл)";}  //  }  //---------------------------------------------------------------------------  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::N6Click(TObject \*Sender)  {  if (darktheme)  {darktheme = false;  N6->Caption="Темная тема(Вкл)";  Form1->Color=clWhite;  Label1->Font->Color=clBlack;  Label2->Font->Color=clBlack;  Label3->Font->Color=clBlack;  Label4->Font->Color=clBlack;  Label5->Font->Color=clBlack;  Label6->Font->Color=clBlack;  Label7->Font->Color=clBlack;  Label8->Font->Color=clBlack;  Label9->Font->Color=clBlack;  Label10->Font->Color=clBlack;  Label11->Font->Color=clBlack; |

Продолжение приложения 1

|  |
| --- |
| Label12->Font->Color=clBlack;  Label13->Font->Color=clBlack;  Label14->Font->Color=clBlack;  Label15->Font->Color=clBlack;  Label16->Font->Color=clBlack;}  else{  darktheme = true;  N6->Caption="Темная тема(Выкл)";  Form1->Color=cl3DDkShadow;  Label1->Font->Color=clWhite;  Label2->Font->Color=clWhite;  Label3->Font->Color=clWhite;  Label4->Font->Color=clWhite;  Label5->Font->Color=clWhite;  Label6->Font->Color=clWhite;  Label7->Font->Color=clWhite;  Label8->Font->Color=clWhite;  Label9->Font->Color=clWhite;  Label10->Font->Color=clWhite;  Label11->Font->Color=clWhite;  Label12->Font->Color=clWhite;  Label13->Font->Color=clWhite;  Label14->Font->Color=clWhite;  Label15->Font->Color=clWhite;  Label16->Font->Color=clWhite;}  }  //---------------------------------------------------------------------------  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Button3Click(TObject \*Sender)  {  if (Button1->Visible) {  int limit = numconvert(AnsiString(Edit1 -> Text).c\_str());  if (limit > 0)  limitdictionary = true;  else  limitdictionary = false;  for (int i=0; i < 3; i++) {  StringGrid1 -> Cols[i]->Clear();  StringGrid1 -> Rows[i]->Clear();  }  int step = numconvert(AnsiString(Edit2 -> Text).c\_str());  int counter;  wstring memobuffer;  counter=0;  Memo2 -> Text="";  vector <node> ans = lz78(Memo1 -> Text.c\_str(), limit);  string val;  wstring wval;  if (step<=ans.size()) {  for (int i=0;i < step; i++)  {  StringGrid1 -> Cells[0][counter] = i+1;  StringGrid1 -> Cells[1][counter] = ans[i].pos;  StringGrid1 -> Cells[2][counter] = ans[i].next;  if (bettercompression){  memobuffer+=wchar\_t(ans[i].pos+1);// 0 для строки в c++ считается концом ввода поэтому +1  memobuffer+=ans[i].next; // так же кодируем цифры наподобие 13 в один символ для еще большего сжатия |

Продолжение приложения 1

|  |
| --- |
| }  else{  val = to\_string(ans[i].pos);  wval = wstring(val.begin(),val.end());  memobuffer += wval;  memobuffer +=L" ";  memobuffer +=ans[i].next;  }  counter++;  }  Memo2 -> Text=memobuffer.c\_str();  }  else{ShowMessage("Количество шагов указано больше чем их может быть");}  }  /////////////////  if (Button2->Visible) {  if (bettercompression) {  int step = numconvert(AnsiString(Edit2 -> Text).c\_str())\*2;  wstring s = Memo2 -> Text.c\_str(); // берем закодированную строку из 2 панели  vector<node>a; // создаем массив для декодирования  for (int i = 0; i < min(step,(int)s.size()); i += 2) // заполняем массив  a.push\_back({ s[i]-1, s[i + 1] }); // -1 для того чтобы исправить костыль с 0 в строке c++ берем символ позиции и следующим символ строки  Memo3 -> Text =decode(a).c\_str(); // выводим декодированную строку на 3 панель  }  else {  wstring s = Memo2 -> Text.c\_str();  vector<node>a;  int i=0,j = 0;  int step = numconvert(AnsiString(Edit2 -> Text).c\_str());  while (i <s.size() and j<step) {  int cv = 0;  while(i < s.size() && s[i] >= L'0' && s[i] <= L'9')  cv \*= 10, cv += s[i] - L'0', ++i;  ++i;  a.push\_back({ cv, s[i] });  ++i;  j++;  }  Memo3 -> Text =decode(a).c\_str();  }  }  }  //--------------------------------------------------------------------------- |

Продолжение приложения 1

|  |
| --- |
| void \_\_fastcall TForm1::Memo1Change(TObject \*Sender)  {  if (Button1->Visible) {  vector <node> ans = lz78(Memo1 -> Text.c\_str());  Label10->Caption=ans.size();  }  wstring ss = Memo1 -> Text.c\_str();  Label13->Caption=ss.size();  if (Label13->Caption != 0 and Label14->Caption != 0) {  Label16->Caption=Label13->Caption/Label14->Caption;  }  }  //---------------------------------------------------------------------------  void \_\_fastcall TForm1::Memo2Change(TObject \*Sender)  {  if (Button2->Visible and bettercompression) {  wstring s = Memo2 -> Text.c\_str();  Label10->Caption=s.size()\*2;  }  if (Button2->Visible and !bettercompression) {  wstring s = Memo2 -> Text.c\_str();  int i=0,j = 0;  while (i <s.size()) {  int cv = 0;  while(i < s.size() && s[i] >= L'0' && s[i] <= L'9')  cv \*= 10, cv += s[i] - L'0', ++i;  ++i;  ++i;  j++;  }  Label10->Caption=j;  }  wstring ss = Memo2 -> Text.c\_str();  Label14->Caption=ss.size();  if (Label13->Caption != 0 and Label14->Caption != 0) {  Label16->Caption=Label13->Caption/Label14->Caption;  }  }  //---------------------------------------------------------------------------  AnsiString getfilename(AnsiString filename){  AnsiString nm;  Form1->OpenDialog1 -> Title = filename;  if (!Form1->OpenDialog1->Execute()) {  ShowMessage("Не указан файл");  return "";  }  nm = Form1->OpenDialog1 -> FileName;  return nm;  }  void \_\_fastcall TForm1::N7Click(TObject \*Sender)  {  AnsiString nfm;  nfm = getfilename("Укажите файл с SQL - утверждением"); |

Продолжение приложения 1

|  |
| --- |
| if (nfm == "") return;  string line;  ifstream in(nfm.c\_str());  if (in.is\_open())  {  while (getline(in, line)){  Memo1-> Lines -> Add(line.c\_str());  }  }  }  //--------------------------------------------------------------------------- |