StringMaker

Documentație

Student: Vladislav Garaba

Grupa: 221

Data: 17.05.2018

Codurile de mai jos reprezinta parti din implementarea aplicatiei “StringMaker” – proiect efectuat pentru demonstrarea cunostintelor

la disciplina “Programare orientate pe obiecte”, prof. Andrioaia Dragoș.

* Fisierul StringMaker.cpp:

*Utilizare:* Declararea claselor care sunt folosite ulterior in rularea si functionarea aplicatiei, cat si conditiile initiale de rulare, declaratii suplimentare sau includerea functionalitatilor de baza.

*Cod:*

// StringMaker.cpp : Defineste clasele care vor raspunde de comportamentul aplicatiei

//

#include "stdafx.h"

#include "StringMaker.h"

#include "StringMakerDlg.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CStringMakerApp

BEGIN\_MESSAGE\_MAP(CStringMakerApp, CWinApp)

ON\_COMMAND(ID\_HELP, &CWinApp::OnHelp)

END\_MESSAGE\_MAP()

// CStringMakerApp constructorul

CStringMakerApp::CStringMakerApp()

{

// support Restart Manager

m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_RESTART;

// TODO: add construction code here,

// Place all significant initialization in InitInstance

}

// Unicul obiect al apicatiei

CStringMakerApp theApp;

// CStringMakerApp initialization

BOOL CStringMakerApp::InitInstance()

{

// InitCommonControlsEx() is required on Windows XP if an application

// manifest specifies use of ComCtl32.dll version 6 or later to enable

// visual styles. Otherwise, any window creation will fail.

INITCOMMONCONTROLSEX InitCtrls;

InitCtrls.dwSize = sizeof(InitCtrls);

// Set this to include all the common control classes you want to use

// in your application.

InitCtrls.dwICC = ICC\_WIN95\_CLASSES;

InitCommonControlsEx(&InitCtrls);

CWinApp::InitInstance();

// Create the shell manager, in case the dialog contains

// any shell tree view or shell list view controls.

CShellManager \*pShellManager = new CShellManager;

// Activate "Windows Native" visual manager for enabling themes in MFC controls

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));

// Standard initialization

// If you are not using these features and wish to reduce the size

// of your final executable, you should remove from the following

// the specific initialization routines you do not need

// Change the registry key under which our settings are stored

// TODO: You should modify this string to be something appropriate

// such as the name of your company or organization

SetRegistryKey(\_T("Local AppWizard-Generated Applications"));

CStringMakerDlg dlg;

m\_pMainWnd = &dlg;

INT\_PTR nResponse = dlg.DoModal();

if (nResponse == IDOK)

{

// TODO: Place code here to handle when the dialog is

// dismissed with OK

}

else if (nResponse == IDCANCEL)

{

// TODO: Place code here to handle when the dialog is

// dismissed with Cancel

}

else if (nResponse == -1)

{

TRACE(traceAppMsg, 0, "Warning: dialog creation failed, so application is terminating unexpectedly.\n");

TRACE(traceAppMsg, 0, "Warning: if you are using MFC controls on the dialog, you cannot #define \_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS.\n");

}

// Delete the shell manager created above.

if (pShellManager != nullptr)

{

delete pShellManager;

}

#if !defined(\_AFXDLL) && !defined(\_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS)

ControlBarCleanUp();

#endif

// Since the dialog has been closed, return FALSE so that we exit the

// application, rather than start the application's message pump.

return FALSE;

* Fisierul StringMakerDlg.cpp:

*Utilizare:* Contine toate implementarile facete pe forma principala. Sunt implementate 2 clase personale care sunt utilizate pentru formatarea textului din Edit Text Box pentru input, fiind ulterior afisat in Edit Text Box pentru output. Mai multe detalii se gaseste in documentatia codului.

*Cod:*

// StringMakerDlg.cpp : implementation file

//

#include "stdafx.h"

#include "StringMaker.h"

#include "StringMakerDlg.h"

#include "afxdialogex.h"

#include <string>

#include <string.h>

#include <iostream>

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CAboutDlg dialog pentru Dialogul "About".

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// Dialog Data

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

// Implementation

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

/\*

Implementarea clasei ajutatoare pentru encriptarea textului.

\*/

class EncryptionHelper

{

private:

// Text variable pentru encryption.

std::string text;

// Destructorul clasei care este privat.

~EncryptionHelper()

{

this->text = '/0';

}

public:

// Constructorul clasei.

EncryptionHelper(std::string text)

{

this->text = text;

}

// Functia de encriptare a textului.

std::string encrypt()

{

if (text.length() < 1)

return this->text;

for (int i = 0; (i < this->text.length() && this->text[i] != '\0'); i++)

this->text[i] = this->text[i] - 25;

return this->text;

}

};

/\*

Declararea clasei proprii unde au fost incluse toate functiile

statice folosite pentru procesarea textului.

\*/

class StringRemakers

{

public:

// Functia de inversare a textului.

static std::string reverse(std::string str)

{

if (str.length() < 1)

return str;

int len = str.length();

// Se interschimba elementul din ultima pozitie cu cel din prima pozitie.

// Se mareste prima pozitie interschimbata cu ultima pozitie, decrementata cu o unitate si aceasta.

// Se repeta pana la jumatatea vectorului.

for (int i = 0; i < len/2; i++)

{

str.at(i) = str.at(i) ^ str.at(len - 1 - i);

str.at(len - 1 - i) = str.at(i) ^ str.at(len - 1 - i);

str.at(i) = str.at(i) ^ str.at(len - 1 - i);

}

return str;

}

// Metoda utilizata pentru compresarea textului.

static std::string compression(std::string str)

{

if (str.length() < 1)

return str;

int len = str.length();

// Crearea unui string nou.

std::string newStr;

for (int i = 0; i < len; i++)

{

// Se scrie elementul in noul string.

char compressedChar = str.at(i);

if (compressedChar != '@')

{

newStr+= compressedChar;

}

else

{

return "INVALID INPUT! No \"@\" symbol permited.";

}

int index = i+1;

// In caz ca se repeta elementul dat, se umara aparitia lui.

if (index < len)

{

int charsInARow =0;

while (index<len && str.at(index) == compressedChar)

{

charsInARow++;

index++;

}

// Dupa numararea caracterelor care se repeta, ele sunt introduse in noul stirng.

if (charsInARow)

{

newStr += "@"+std::to\_string(charsInARow+1);

}

i += charsInARow;

}

}

return newStr;

}

// Metoda utilizata pentru decompresarea textului.

static std::string decompression(std::string str)

{

if (str.length() < 1)

return str;

int len = str.length();

std::string newStr;

for (int i = 0; i < len;)

{

char checkedChar = str.at(i);

// Se cauta elementul '@' care e simbolul unic a compresarii.

if (checkedChar == '@')

{

if (!isdigit(str.at(i + 1)))

{

return "INVALID INPUT! No digit detected after \"@\" symbol.";

}

char previousChar = str.at(i - 1);

int times = atoi(&str.at(i + 1));

for (int j = 1; j < times; j++)

{

newStr += previousChar;

}

i += 2;

}

else

{

newStr += checkedChar;

i++;

}

}

return newStr;

}

// Metoda pentru schimbarea case-lui literelor.

static std::string changeCase(std::string str)

{

if (str.length() < 1)

return str;

std::string lower= "qwertyuiopasdfghjklzxcvbnm";

std::string upper = "QWERTYUIOPLKJHGFDSAZXCVBNM";

int len = str.length();

for (int i = 0; i < len; i++)

{

if (lower.find(str.at(i)) != -1)

{

str.at(i) = str.at(i)-32;

continue;

}

if (upper.find(str.at(i)) != -1)

{

str.at(i) = str.at(i) + 32;

}

}

return str;

}

// Inversarea cuvintelor in text.

static std::string reverseWords(std::string str)

{

if (str.length() < 1)

return str;

std::string newStr;

for (int i = str.length()-1; i >=0;)

{

if (isalpha(str.at(i)) || isdigit(str.at(i)))

{

int index = i;

//Se numara urmatoarele litere a stringului si creste valoarea cat timp e litera/cifra.

while (index >= 0 && (isalpha(str.at(index)) || isdigit(str.at(index))))

{

index--;

}

//Se introduce cuvantul detectat in noul string.

for (int j = index+1; j <= i; j++)

newStr += str.at(j);

i = index;

}

else

{

int index = i;

//

// Se cauta toate simbolurile consecutive.

while (index >= 0 && (!isalpha(str.at(index)) && !isdigit(str.at(index))))

{

index--;

}

// Se introduc in noul stirng.

for (int j = index + 1; j <= i; j++)

newStr += str.at(j);

i = index;

}

}

return newStr;

}

// Metoda pentru numerotarea vocalelor.

static std::string countVowels(std::string str)

{

if (str.length() < 1)

return str;

// Se declara un vector container al vocalelor.

std::string vowels("aeiou");

int len = str.length();

int countVowels = 0;

//Se cauta vocalele in text si se numara.

for (int i = 0; i < len; i++)

{

if (vowels.find(str.at(i))!=-1)

{

countVowels++;

}

}

return "Text has " + std::to\_string(countVowels) + " vowels.";

}

// Metoda pentru numararea consoanelor.

static std::string countConsons(std::string str)

{

if (str.length() < 1)

return str;

// Se declara un vector container al consoanelor.

std::string consons("qwrtyplkjhgfdszxcvbnm");

int len = str.length();

int countConsons = 0;

//Se cauta consoanele in text si se numara.

for (int i = 0; i < len; i++)

{

if (consons.find(str.at(i)) != -1)

{

countConsons++;

}

}

return "Text has " + std::to\_string(countConsons) + " consons.";

}

// Metoda de numarare a cuvintelor se face pe baza spatiilor gasite intre cuvinte.

static std::string countWords(std::string str)

{

// Se verifica daca textul nu este null.

if (str.length() < 1)

return str;

int nSpaces = 0;

unsigned int i = 0;

// Sare peste primele spatii.

while (isspace(str.at(i)))

i++;

// Se numara fiecare spatiu.

for (; i < str.length(); i++)

{

if (isspace(str.at(i)))

{

nSpaces++;

// Se sare peste spatiile multiple.

while (isspace(str.at(i++)) && i < str.length());

}

}

// Daca ultimul element e spatiu, atunci se decrementeaza variabila nSpaces.

if (isspace(str.at(str.length() - 1)))

nSpaces--;

return "Text has "+std::to\_string(nSpaces + 1)+" words.";

}

// Metoda de encriptare a textului.

static std::string encrypt(std::string str)

{

EncryptionHelper \*encryptor = new EncryptionHelper(str);

return encryptor->encrypt();

}

// Metoda de decriptare a textului.

static std::string decrypt(std::string str)

{

if (str.length() < 1)

return str;

for (int i = 0; (i < str.length() && str[i] != '\0'); i++)

str[i] = str[i] + 25;

return str;

}

};

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

// Focusarea dialogurilor de pornire / oprire a programului.

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

/\*

Initialiazarea variabilelor folosite in acest dialog.

\*/

CStringMakerDlg::CStringMakerDlg(CWnd\* pParent /\*=nullptr\*/)

: CDialogEx(IDD\_STRINGMAKER\_DIALOG, pParent)

, userOutput(\_T("Output goes here."))

, userInput(\_T("Enter your text here."))

, instructionOutput(\_T(""))

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

/\*

Asignarea variabilelor cu elementele formei.

De asemenea declararea unui font, folosit pentru textul static din titlu.

\*/

void CStringMakerDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_TITLE\_STRING, m\_label);

CFont font;

LOGFONT lf;

memset(&lf, 0, sizeof(LOGFONT)); // zero out structure

lf.lfHeight = 21; // request a 12-pixel-height font

\_tcsncpy\_s(lf.lfFaceName, LF\_FACESIZE,

\_T("Tahoma"), 21); // request a face name "Arial"

VERIFY(font.CreateFontIndirect(&lf));

m\_label.SetFont(&font, TRUE);

DDX\_Text(pDX, IDC\_EDIT1, userInput);

DDX\_Text(pDX, IDC\_EDIT2, userOutput);

}

/\*

Definirea metodelor de manipulare a elementelor de pe forma.

\*/

BEGIN\_MESSAGE\_MAP(CStringMakerDlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_CBN\_SELCHANGE(NULL, &CStringMakerDlg::OnCbnSelchangeCombo1)

ON\_BN\_CLICKED(IDC\_RADIO7, &CStringMakerDlg::OnBnClickedRadio7)

ON\_BN\_CLICKED(IDC\_RADIO11, &CStringMakerDlg::OnBnClickedRadio11)

ON\_BN\_CLICKED(IDC\_RADIO3, &CStringMakerDlg::OnBnClickedRadio3)

ON\_BN\_CLICKED(IDC\_RADIO2, &CStringMakerDlg::OnBnClickedRadio2)

ON\_BN\_CLICKED(IDC\_RADIO10, &CStringMakerDlg::OnBnClickedRadio10)

ON\_BN\_CLICKED(IDOK, &CStringMakerDlg::OnBnClickedOk)

ON\_BN\_CLICKED(IDCANCEL, &CStringMakerDlg::OnBnClickedCancel)

ON\_BN\_CLICKED(IDC\_BUTTON2, &CStringMakerDlg::OnBnClickedButton2)

ON\_EN\_CHANGE(IDC\_EDIT1, &CStringMakerDlg::OnEnChangeEdit1)

ON\_EN\_CHANGE(IDC\_EDIT2, &CStringMakerDlg::OnEnChangeEdit2)

ON\_BN\_CLICKED(IDC\_RADIO9, &CStringMakerDlg::OnBnClickedRadio9)

ON\_BN\_CLICKED(IDC\_RADIO8, &CStringMakerDlg::OnBnClickedRadio8)

ON\_BN\_CLICKED(IDC\_RADIO4, &CStringMakerDlg::OnBnClickedRadio4)

ON\_BN\_CLICKED(IDC\_RADIO5, &CStringMakerDlg::OnBnClickedRadio5)

ON\_BN\_CLICKED(IDC\_RADIO6, &CStringMakerDlg::OnBnClickedRadio6)

ON\_BN\_CLICKED(IDC\_MFCBUTTON2, &CStringMakerDlg::OnBnClickedMfcbutton2)

END\_MESSAGE\_MAP()

/\*

Initializarea formei.

\*/

BOOL CStringMakerDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// Selectarea butonului implicit la pornirea formei.

CWnd::CheckRadioButton(IDC\_RADIO2, IDC\_RADIO11, IDC\_RADIO7);

option = radioButtonCol\_1Row1;

// IDM\_ABOUTBOX must be in the system command range.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Set the icon for this dialog. The framework does this automatically

// when the application's main window is not a dialog

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here

return TRUE; // return TRUE unless you set the focus to a control

}

/\*

Forma "about". Nu se utilizeaza.

\*/

void CStringMakerDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// If you add a minimize button to your dialog, you will need the code below

// to draw the icon. For MFC applications using the document/view model,

// this is automatically done for you by the framework.

void CStringMakerDlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // device context for painting

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Center icon in client rectangle

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

// The system calls this function to obtain the cursor to display while the user drags

// the minimized window.

HCURSOR CStringMakerDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CStringMakerDlg::OnCbnSelchangeCombo1()

{

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio7()

{

option = radioButtonCol\_1Row1;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//change instruction text

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option counts all the words from the text and shows their number."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio11()

{

option = radioButtonCol\_1Row5;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option decrypts the input text with the default key and shows its content."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio3()

{

option = radioButtonCol\_2Row2;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option compresses the multiple symbols which are in a row (more than 2). '@' symbol is forbidden.\nIt generates an '@' symbol followed by a number which represents how many symbols were in a row.\nExample: \"zoo\" is converted in \"zo@2\" where \"oo\" is converted into \"o@2\" which means there were 2 'o' symbol in a row."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio2()

{

option = radioButtonCol\_2Row1;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option switches letters cases. Upper case letter becomes lower case and vice versa."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio10()

{

option = radioButtonCol\_1Row4;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option encrypts the input text with a built-in key and shows its content."));

}

/\*

Codul pentru butonul "Make it".

Contine un switch pentru schimbarea optiunii, in dependenta de butonul radio selectat.

Dupa asta, dupa care reformateaza textul si il afiseaza in Edit Box-ul de la output.

\*/

void CStringMakerDlg::OnBnClickedOk()

{

UpdateData();

GetDlgItem(IDC\_EDIT2)->EnableWindow(TRUE);

// Transforming input to userOutput

switch (option)

{

case radioButtonCol\_1Row1:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::countWords(in)).c\_str();

break;

}

case radioButtonCol\_1Row2:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::countVowels(in)).c\_str();

break;

}

case radioButtonCol\_1Row3:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::countConsons(in)).c\_str();

break;

}

case radioButtonCol\_1Row4:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::encrypt(in)).c\_str();

break;

}

case radioButtonCol\_1Row5:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::decrypt(in)).c\_str();

break;

}

case radioButtonCol\_2Row1:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::changeCase(in)).c\_str();

break;

}

case radioButtonCol\_2Row2:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::compression(in)).c\_str();

break;

}

case radioButtonCol\_2Row3:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::decompression(in)).c\_str();

break;

}

case radioButtonCol\_2Row4:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::reverse(in)).c\_str();

break;

}

case radioButtonCol\_2Row5:

{

CT2CA pszConvertedAnsiString(userInput);

std::string in(pszConvertedAnsiString);

userOutput = (StringRemakers::reverseWords(in)).c\_str();

break;

}

break;

default:

break;

}

GetDlgItem(IDC\_MFCBUTTON2)->EnableWindow(TRUE);

GetDlgItem(IDC\_EDIT2)->SetWindowTextW(userOutput);

}

/\*

Butonul de inchidere a programului.

\*/

void CStringMakerDlg::OnBnClickedCancel()

{

CDialogEx::OnCancel();

}

/\*

Nu se utilizeaza.

@deprecated

\*/

void CStringMakerDlg::OnBnClickedCommandexit()

{

// TODO: Add your control notification handler code here

//CDialogEx::On

}

void CStringMakerDlg::OnBnClickedButton2()

{

// TODO: Add your control notification handler code here

CDialogEx::OnCancel();

}

/\*

Codul pentru Text Box-ul de input.

\*/

void CStringMakerDlg::OnEnChangeEdit1()

{

// TODO: If this is a RICHEDIT control, the control will not

// send this notification unless you override the CDialogEx::OnInitDialog()

// function and call CRichEditCtrl().SetEventMask()

// with the ENM\_CHANGE flag ORed into the mask.

}

/\*

Codul pentru Text Box-ul de output.

\*/

void CStringMakerDlg::OnEnChangeEdit2()

{

// TODO: If this is a RICHEDIT control, the control will not

// send this notification unless you override the CDialogEx::OnInitDialog()

// function and call CRichEditCtrl().SetEventMask()

// with the ENM\_CHANGE flag ORed into the mask.

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio9()

{

option = radioButtonCol\_1Row3;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//Editam sectiunea pentru instrctiune.

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option counts all the consons from the input text and shows their number."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio8()

{

option = radioButtonCol\_1Row2;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option counts all the vowels from the input text and shows thier number."));

// TODO: Add your control notification handler code here

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio4()

{

option = radioButtonCol\_2Row3;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option will decompress an already compressed text.\nNote: Characters after '@' symbol are not allowed. Also check out \"Compression text\" instruction for more details."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio5()

{

option = radioButtonCol\_2Row4;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option reverses the text and shows its reversed version."));

}

/\*

Codul pentru butonul radio.

\*/

void CStringMakerDlg::OnBnClickedRadio6()

{

option = radioButtonCol\_2Row5;

GetDlgItem(IDOK)->EnableWindow(TRUE);

GetDlgItem(IDOK)->BringWindowToTop();

//

GetDlgItem(IDC\_STATIC3)->SetWindowText(TEXT("Instruction: This option reverses words from the text.\nExample: \"I do coding.\" becomes \".coding do I\""));

}

/\*

Codul pentru butonul de schimb al textului.

Acest buton schimba codul de la input, cu cel de la output, si sterge textul de la output.

\*/

void CStringMakerDlg::OnBnClickedMfcbutton2()

{

// TODO: Add your control notification handler code here

UpdateData();

userInput = userOutput;

userOutput = "";

GetDlgItem(IDC\_EDIT1)->SetWindowTextW(userInput);

GetDlgItem(IDC\_EDIT2)->SetWindowTextW(userOutput);

//userInput(\_T("Enter your text here."));

}

* Fisierul StringMakerDlg.h:

*Utilizare:* Se rezerva memoria pentru variabilele care vor fi folosite ulterior in dialog.

*Cod:*

// StringMakerDlg.h : header file

// Declararea tuturor claselor sau variabilelor care vor fi folosite ulterior.

#pragma once

// CStringMakerDlg dialog

class CStringMakerDlg : public CDialogEx

{

// Construction

public:

CStringMakerDlg(CWnd\* pParent = nullptr); // standard constructor

// Dialog Data

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_STRINGMAKER\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

// Implementation

protected:

HICON m\_hIcon;

// Generated message map functions

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

CStatic m\_label;

afx\_msg void OnCbnSelchangeCombo1();

afx\_msg void OnBnClickedRadio7();

afx\_msg void OnBnClickedRadio11();

afx\_msg void OnBnClickedRadio3();

afx\_msg void OnBnClickedRadio2();

afx\_msg void OnBnClickedRadio10();

afx\_msg void OnBnClickedOk();

afx\_msg void OnBnClickedCancel();

afx\_msg void OnBnClickedCommandexit();

afx\_msg void OnBnClickedButton2();

afx\_msg void OnEnChangeEdit1();

afx\_msg void OnEnChangeEdit2();

afx\_msg void OnBnClickedRadio9();

// Variablila pentru input

CString userInput;

// Enumeratia pentru alegerea optiunii de formatare a textului.

enum radioButtons

{

radioButtonCol\_1Row1,

radioButtonCol\_1Row2,

radioButtonCol\_1Row3,

radioButtonCol\_1Row4,

radioButtonCol\_1Row5,

radioButtonCol\_2Row1,

radioButtonCol\_2Row2,

radioButtonCol\_2Row3,

radioButtonCol\_2Row4,

radioButtonCol\_2Row5

};

// Variabila a enumeratiei.

radioButtons option;

afx\_msg void OnBnClickedRadio8();

afx\_msg void OnBnClickedRadio4();

afx\_msg void OnBnClickedRadio5();

afx\_msg void OnBnClickedRadio6();

// This goes to show output to user

CString userOutput;

CString instructionOutput;

afx\_msg void OnBnClickedMfcbutton2();

afx\_msg void OnBnClickedButton1();

};

* Fisierul targetver.h:

*Utilizare:* Definirea platformei superioare windows pe care poate rula aplicatia data.

*Cod:*

#pragma once

// Including SDKDDKVer.h defines the highest available Windows platform.

// If you wish to build your application for a previous Windows platform, include WinSDKVer.h and

// set the \_WIN32\_WINNT macro to the platform you wish to support before including SDKDDKVer.h.

#include <SDKDDKVer.h>

* Fisierul StringMaker.h:

*Utilizare:* Fisierul header principal. Se declara cele mai fundamentale clase / variabile.

*Cod:*

// StringMaker.h : main header file for the TextMaker application

//

#pragma once

#ifndef \_\_AFXWIN\_H\_\_

#error "include 'stdafx.h' before including this file for PCH"

#endif

#include "resource.h" // main symbols

// CStringMakerApp:

// See StringMaker.cpp for the implementation of this class

//

class CStringMakerApp : public CWinApp

{

public:

CStringMakerApp();

// Overrides

public:

virtual BOOL InitInstance();

// Implementation

DECLARE\_MESSAGE\_MAP()

};

extern CStringMakerApp theApp;