#ifndef HEADER

#define HEADER

#include <string>

#include <iostream>

using namespace std;

class CRealEstate

{

static int s\_nPrice;

protected:

int m\_nNumbersRooms;

int m\_nTotalArea;

int m\_nPrice;

string m\_sType;

string m\_sAddress;

string m\_sOwner;

public:

int nCountRP;

int nCountNRP;

bool isSearchAddress();

bool isSearchOwner();

bool isSearchNumbersOfRooms();

bool isSearchTotalArea();

bool isSearchPrice();

void vSetPrice(int \_nPrice);

void vShowPrice();

void vPrintHeaders();

void vAddObjectToStringGrid(bool \_isRP);

void vReadFile();

void vDeleteObject();

string sGetOwner();

};

class CResidentialPremises : public CRealEstate

{

private:

int m\_nKitchenArea;

int m\_nRoomsArea;

string m\_sRepairCondition;

public:

int nGetNumbersRooms();

int nGetTotalArea();

bool isSearchKitchenArea();

bool isSearchRoomsArea();

bool isSearchRepairCondition();

bool isDefinitionSquareMax(CResidentialPremises &tmpobj);

bool isDefinitionSquareMin(CResidentialPremises &tmpobj);

void vSetValueForRP(int);

void vSortByAddress(CResidentialPremises &tmpobj);

void vSortByNumbersRooms(CResidentialPremises &tmpobj);

void vSwapForRP(CResidentialPremises &tmpobj);

void vPrintRPInForm(int);

string sGetAddressRP();

};

class CNonResidentialPremises : public CRealEstate

{

private:

string m\_sPurpose;

public:

int nGetTotalArea();

int nGetNumbersRooms();

bool isSearchPurpose();

bool isDefinitionSquareMax(CNonResidentialPremises &tmpobj);

bool isDefinitionSquareMin(CNonResidentialPremises &tmpobj);

void vSetValueForNRP(int);

void vSortByPurpose(CNonResidentialPremises &tmpobj);

void vSortByAddress(CNonResidentialPremises &tmpobj);

void vSortByNumbersRooms(CNonResidentialPremises &tmpobj);

void vSwapForNRP(CNonResidentialPremises &tmpobj);

void vPrintNRPInForm(int);

string sGetAddressNRP();

};

#endif

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAa

#include "Unit1.h"

#include "File2.h"

#include <fstream>

#include <iostream>

#include <cctype>

using namespace std;

//ïåðåâ³ðêà ÷è çíàéäåíà àäðåñà îá'ºêòà

bool CRealEstate :: isSearchAddress()

{

if(this->m\_sAddress == RES->Edit1->Text.c\_str())

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíèé âëàñíèê îá'ºêòà

bool CRealEstate :: isSearchOwner()

{

if(this->m\_sOwner == RES->Edit1->Text.c\_str())

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíà ê³ëüê³ñòü ê³ìíàò îá'ºêòà

bool CRealEstate :: isSearchNumbersOfRooms()

{

if(this->m\_nNumbersRooms == StrToInt(RES->Edit1->Text))

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíà çàãàëüíà ïëîùà îá'ºêòà

bool CRealEstate :: isSearchTotalArea()

{

if(this->m\_nTotalArea == StrToInt(RES->Edit1->Text))

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíà ö³íà îá'ºêòà

bool CRealEstate :: isSearchPrice()

{

if(this->m\_nPrice == StrToInt(RES->Edit1->Text))

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíà ïëîùà êóõí³ îá'ºêòà

bool CResidentialPremises :: isSearchKitchenArea()

{

if(this->m\_nKitchenArea == StrToInt(RES->Edit1->Text))

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíà ïëîùà ê³ìíàò îá'ºêòà

bool CResidentialPremises :: isSearchRoomsArea()

{

if(this->m\_nRoomsArea == StrToInt(RES->Edit1->Text))

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíèé ñòàí ðåìîíòó îá'ºêòà

bool CResidentialPremises :: isSearchRepairCondition()

{

if(this->m\_sRepairCondition == RES->Edit1->Text.c\_str())

return true;

else

return false;

}

//ïåðåâ³ðêà ÷è çíàéäåíå ö³ëüîâå ïðèçíà÷åííÿ îá'ºêòà

bool CNonResidentialPremises :: isSearchPurpose()

{

if(this->m\_sPurpose == RES->Edit1->Text.c\_str())

return true;

else

return false;

}

//ôóíêö³ÿ ç÷èòóâàííÿ ³ç ôàéëó òà çàïèñóâàííÿ ó StringGrid

void CRealEstate :: vReadFile()

{

ifstream fin("Read.txt");

char chInformation[200];

bool isInfo = true;

nCountRP = 0;

nCountNRP = 0;

if(isInfo)

{

isInfo = false;

fin.getline(chInformation,200) ;

}

int c = 0, i = 0, count , j=0;

char ch;

bool isFlag;

bool isOne = true;

char \*mas;

do

{

if (fin.eof())

break;

c = 0;

i = 0;

isOne = true;

while((ch = fin.get())!='|');

mas = new char[4];

while((ch = fin.get()) != '|')

{

if(i < 3)

{

if(isOne)

{

if(ch == 'R')

isFlag = true;

else

isFlag = false;

isOne = false;

}

mas[i] = ch;

++i;

mas[i] = '\0';

}

else

continue;

}

RES->StringGrid1->Cells[c][j+1] = mas;

++c;

delete []mas;

char \*mas = new char[10];

i = 0;

while((ch = fin.get())!='|')

{

if(i < 7)

{

mas[i] = ch;

++i;

mas[i] = '\0';

}

else

continue;

}

RES->StringGrid1->Cells[c][j+1] = mas;

++c;

delete []mas;

mas = new char[10];

i = 0;

while((ch = fin.get())!='|')

{

if(i < 7 )

{

mas[i] = ch;

++i;

mas[i] = '\0';

}

else

continue;

}

RES->StringGrid1->Cells[c][j+1] = mas;

++c;

delete []mas;

mas = new char[10];

fin >> count;

RES->StringGrid1->Cells[c][j+1] = IntToStr(count);

++c;

while((ch = fin.get())!='|');

fin >> count;

RES->StringGrid1->Cells[c][j+1] = IntToStr(count);

++c;

while((ch = fin.get())!='|');

if(isFlag)

{

fin >> count ;

RES->StringGrid1->Cells[c][j+1] = IntToStr(count);

++c;

while((ch = fin.get())!='|') ;

fin >> count ;

RES->StringGrid1->Cells[c][j+1] = IntToStr(count);

++c;

while((ch = fin.get())!='|');

}

else

{

++c;

while((ch = fin.get())!='|') ;

++c;

while((ch = fin.get())!='|');

}

i = 0;

if(isFlag)

{

while((ch = fin.get())!='|')

{

if(i < 7 && isalpha(ch))

{

mas[i] = ch;

++i;

mas[i] = '\0';

}

else

continue;

}

RES->StringGrid1->Cells[c][j+1] = mas;

++c;

delete []mas;

}

else

{

while((ch = fin.get())!='|');

++c;

}

mas = new char[10];

i = 0;

while((ch = fin.get())!='|')

{

if(i < 7)

{

mas[i] = ch;

++i;

mas[i] = '\0';

}

else

continue;

}

RES->StringGrid1->Cells[c][j+1] = mas;

++c;

i = 0;

fin >> count;

while((ch = fin.get())!='|');

RES->StringGrid1->Cells[c][j+1] = IntToStr(count);

delete []mas;

if(RES->StringGrid1->Cells[0][j+1]=="RP ")

++nCountRP;

else if(RES->StringGrid1->Cells[0][j+1]=="NRP")

++nCountNRP;

++j;

fin.getline(chInformation,200) ;

}while(1);

fin.close();

}

//çàïîâíåííÿ æèòëîâèõ ïðèì³ùåíü

void CResidentialPremises :: vSetValueForRP(int \_nRows)

{

m\_sType = (RES->StringGrid1->Cells[0][\_nRows + 1]).c\_str();

m\_sAddress = (RES->StringGrid1->Cells[1][\_nRows + 1]).c\_str();

m\_sOwner = (RES->StringGrid1->Cells[2][\_nRows + 1]).c\_str();

m\_nNumbersRooms = StrToInt(RES->StringGrid1->Cells[3][\_nRows + 1]);

m\_nTotalArea = StrToInt(RES->StringGrid1->Cells[4][\_nRows + 1]);

m\_nKitchenArea = StrToInt(RES->StringGrid1->Cells[5][\_nRows + 1]);;

m\_nRoomsArea = StrToInt(RES->StringGrid1->Cells[6][\_nRows + 1]);;

m\_sRepairCondition = (RES->StringGrid1->Cells[7][\_nRows + 1]).c\_str();

m\_nPrice = StrToInt(RES->StringGrid1->Cells[9][\_nRows + 1]);

}

//çàïîâíåííÿ íåæèòëîâèõ ïðèì³ùåíü

void CNonResidentialPremises :: vSetValueForNRP(int \_nRows)

{

m\_sType = (RES->StringGrid1->Cells[0][\_nRows + 1]).c\_str();

m\_sAddress = (RES->StringGrid1->Cells[1][\_nRows + 1]).c\_str();

m\_sOwner = (RES->StringGrid1->Cells[2][\_nRows + 1]).c\_str();

m\_nNumbersRooms = StrToInt(RES->StringGrid1->Cells[3][\_nRows + 1]);

m\_nTotalArea = StrToInt(RES->StringGrid1->Cells[4][\_nRows + 1]);

m\_sPurpose = (RES->StringGrid1->Cells[8][\_nRows + 1]).c\_str();

m\_nPrice = StrToInt(RES->StringGrid1->Cells[9][\_nRows + 1]);

}

//ïîâåðåííÿ ê³ëüêîñò³ ê³ìíàò

int CNonResidentialPremises :: nGetNumbersRooms()

{

return m\_nNumbersRooms;

}

//ïîâåðåííÿ àäðåñè

string CNonResidentialPremises :: sGetAddressNRP()

{

return m\_sAddress;

}

//ïîâåðåííÿ çàãàëüíî¿ ïëîù³

int CNonResidentialPremises :: nGetTotalArea()

{

return m\_nTotalArea;

}

//ïîð³âíÿííÿ çà ö³ëüîâèì ïðèçíà÷åííÿì

void CNonResidentialPremises :: vSortByPurpose(CNonResidentialPremises &tmpobj)

{

if(m\_sPurpose > tmpobj.m\_sPurpose)

vSwapForNRP(tmpobj);

}

//ïîð³âíÿííÿ çà àäðåñîþ

void CNonResidentialPremises :: vSortByAddress(CNonResidentialPremises &tmpobj)

{

if(m\_sAddress > tmpobj.m\_sAddress)

vSwapForNRP(tmpobj);

}

//ïîð³âíÿííÿ çà ê³ëüê³ñòþ ê³ìíàò

void CNonResidentialPremises :: vSortByNumbersRooms(CNonResidentialPremises &tmpobj)

{

if(m\_nNumbersRooms > tmpobj.m\_nNumbersRooms)

vSwapForNRP(tmpobj);

}

//îáì³í íåæèòëîâèõ îá'ºêò³â

void CNonResidentialPremises :: vSwapForNRP(CNonResidentialPremises &tmpobj)

{

string sTmpType, sTmpAddress, sTmpOwner, sTmpPurpose;

int nTmpNumbersRooms, nTmpTotalArea, nTmpPrice;

sTmpType = tmpobj.m\_sType;

tmpobj.m\_sType = this->m\_sType;

this->m\_sType = sTmpType;

sTmpAddress = tmpobj.m\_sAddress;

tmpobj.m\_sAddress = this->m\_sAddress;

this->m\_sAddress = sTmpAddress;

sTmpOwner = tmpobj.m\_sOwner;

tmpobj.m\_sOwner = this->m\_sOwner;

this->m\_sOwner = sTmpOwner;

nTmpNumbersRooms = tmpobj.m\_nNumbersRooms;

tmpobj.m\_nNumbersRooms = this->m\_nNumbersRooms;

this->m\_nNumbersRooms = nTmpNumbersRooms;

nTmpTotalArea = tmpobj.m\_nTotalArea;

tmpobj.m\_nTotalArea = this->m\_nTotalArea;

this->m\_nTotalArea = nTmpTotalArea;

sTmpPurpose = tmpobj.m\_sPurpose;

tmpobj.m\_sPurpose = this->m\_sPurpose;

this->m\_sPurpose = sTmpPurpose;

nTmpPrice = tmpobj.m\_nPrice;

tmpobj.m\_nPrice = this->m\_nPrice;

this->m\_nPrice = nTmpPrice;

}

//âèâ³ä íåæèòëîâîãî îá'ºêòà ó StringGrid

void CNonResidentialPremises :: vPrintNRPInForm(int \_nRows)

{

RES->StringGrid1->Cells[0][\_nRows+1] = m\_sType.c\_str();

RES->StringGrid1->Cells[1][\_nRows+1] = m\_sAddress.c\_str();

RES->StringGrid1->Cells[2][\_nRows+1] = m\_sOwner.c\_str();

RES->StringGrid1->Cells[3][\_nRows+1] = IntToStr(m\_nNumbersRooms);

RES->StringGrid1->Cells[4][\_nRows+1] = IntToStr(m\_nTotalArea);

RES->StringGrid1->Cells[8][\_nRows+1] = m\_sPurpose.c\_str();

RES->StringGrid1->Cells[9][\_nRows+1] = IntToStr(m\_nPrice);

}

//ïåðåâ³ðêà çíàéäåíî¿ ìàêñèìàëüíî¿ ïëîù³

bool CNonResidentialPremises :: isDefinitionSquareMax(CNonResidentialPremises &tmpobj)

{

bool isReplace;

if(m\_nTotalArea < tmpobj.m\_nTotalArea)

isReplace = true;

return isReplace;

}

//ïåðåâ³ðêà çíàéäåíî¿ ì³í³ìàëüíî¿ ïëîù³

bool CNonResidentialPremises :: isDefinitionSquareMin(CNonResidentialPremises &tmpobj)

{

bool isReplace;

if(m\_nTotalArea > tmpobj.m\_nTotalArea)

isReplace = true;

return isReplace;

}

//ïîâåðåííÿ ê³ëüêîñò³ ê³ìíàò

int CResidentialPremises :: nGetNumbersRooms()

{

return m\_nNumbersRooms;

}

//ïîâåðåííÿ àäðåñè

string CResidentialPremises :: sGetAddressRP()

{

return m\_sAddress;

}

//ïîð³âíÿííÿ çà àäðåñîþ

void CResidentialPremises :: vSortByAddress(CResidentialPremises &tmpobj)

{

if(m\_sAddress > tmpobj.m\_sAddress)

vSwapForRP(tmpobj);

}

//ïîð³âíÿííÿ çà ê³ëüê³ñòþ ê³ìíàò

void CResidentialPremises :: vSortByNumbersRooms(CResidentialPremises &tmpobj)

{

if(m\_nNumbersRooms > tmpobj.m\_nNumbersRooms)

vSwapForRP(tmpobj);

}

//îáì³í æèòëîâèõ îá'ºêò³â

void CResidentialPremises :: vSwapForRP(CResidentialPremises &tmpobj)

{

string sTmpType, sTmpAddress, sTmpOwner, sTmpRepairCondition;

int nTmpNumbersRooms, nTmpTotalArea, nTmpPrice, nTmpKitchenArea, nTmpRoomsArea;

sTmpType = tmpobj.m\_sType;

tmpobj.m\_sType = this->m\_sType;

this->m\_sType = sTmpType;

sTmpAddress = tmpobj.m\_sAddress;

tmpobj.m\_sAddress = this->m\_sAddress;

this->m\_sAddress = sTmpAddress;

sTmpOwner = tmpobj.m\_sOwner;

tmpobj.m\_sOwner = this->m\_sOwner;

this->m\_sOwner = sTmpOwner;

nTmpNumbersRooms = tmpobj.m\_nNumbersRooms;

tmpobj.m\_nNumbersRooms = this->m\_nNumbersRooms;

this->m\_nNumbersRooms = nTmpNumbersRooms;

nTmpTotalArea = tmpobj.m\_nTotalArea;

tmpobj.m\_nTotalArea = this->m\_nTotalArea;

this->m\_nTotalArea = nTmpTotalArea;

nTmpKitchenArea= tmpobj.m\_nKitchenArea;

tmpobj.m\_nKitchenArea = this->m\_nKitchenArea;

this->m\_nKitchenArea = nTmpKitchenArea;

nTmpRoomsArea = tmpobj.m\_nRoomsArea;

tmpobj.m\_nRoomsArea = this->m\_nRoomsArea;

this->m\_nRoomsArea = nTmpRoomsArea;

sTmpRepairCondition = tmpobj.m\_sRepairCondition;

tmpobj.m\_sRepairCondition = this->m\_sRepairCondition;

this->m\_sRepairCondition = sTmpRepairCondition;

nTmpPrice = tmpobj.m\_nPrice;

tmpobj.m\_nPrice = this->m\_nPrice;

this->m\_nPrice = nTmpPrice;

}

//âèâ³ä îá'ºêòà ó StringGrid

void CResidentialPremises :: vPrintRPInForm(int \_nRows)

{

RES->StringGrid1->Cells[0][\_nRows+1] = m\_sType.c\_str();

RES->StringGrid1->Cells[1][\_nRows+1] = m\_sAddress.c\_str();

RES->StringGrid1->Cells[2][\_nRows+1] = m\_sOwner.c\_str();

RES->StringGrid1->Cells[3][\_nRows+1] = IntToStr(m\_nNumbersRooms);

RES->StringGrid1->Cells[4][\_nRows+1] = IntToStr(m\_nTotalArea);

RES->StringGrid1->Cells[5][\_nRows+1] = IntToStr(m\_nKitchenArea);

RES->StringGrid1->Cells[6][\_nRows+1] = IntToStr(m\_nRoomsArea);

RES->StringGrid1->Cells[7][\_nRows+1] = m\_sRepairCondition.c\_str();

RES->StringGrid1->Cells[9][\_nRows+1] = IntToStr(m\_nPrice);

}

//ïåðåâ³ðêà çíàéäåíî¿ ìàêñèìàëüíî¿ ïëîù³

bool CResidentialPremises :: isDefinitionSquareMax(CResidentialPremises &tmpobj)

{

bool isReplace;

if(m\_nTotalArea < tmpobj.m\_nTotalArea)

isReplace = true;

return isReplace;

}

//ïåðåâ³ðêà çíàéäåíî¿ ì³í³ìàëüíî¿ ïëîù³

bool CResidentialPremises :: isDefinitionSquareMin(CResidentialPremises &tmpobj)

{

bool isReplace;

if(m\_nTotalArea > tmpobj.m\_nTotalArea)

isReplace = true;

return isReplace;

}

//ïîâåðíÿííÿ çàãàëüíî¿ ïëîù³

int CResidentialPremises :: nGetTotalArea()

{

return m\_nTotalArea;

}

//ïîâåðíåííÿ ³ìåí³ âëàñíèêà

string CRealEstate :: sGetOwner()

{

return m\_sOwner;

}

//äîäàâàííÿ îá'ºêòà

void CRealEstate :: vAddObjectToStringGrid(bool \_isRP)

{

RES->StringGrid1->Visible = true;

RES->StringGrid2->Visible = false;

RES->Button4->Visible = false;

int \_nRows = nCountRP + nCountNRP;

RES->StringGrid1->Cells[0][\_nRows+1] = RES->StringGrid2->Cells[0][1];

RES->StringGrid1->Cells[1][\_nRows+1] = RES->StringGrid2->Cells[1][1];

RES->StringGrid1->Cells[2][\_nRows+1] = RES->StringGrid2->Cells[2][1];

RES->StringGrid1->Cells[3][\_nRows+1] = RES->StringGrid2->Cells[3][1];

RES->StringGrid1->Cells[4][\_nRows+1] = RES->StringGrid2->Cells[4][1];

if(\_isRP)

RES->StringGrid1->Cells[5][\_nRows+1] = RES->StringGrid2->Cells[5][1];

if(\_isRP)

RES->StringGrid1->Cells[6][\_nRows+1] = RES->StringGrid2->Cells[6][1];

if(\_isRP)

RES->StringGrid1->Cells[7][\_nRows+1] = RES->StringGrid2->Cells[7][1];

if(!\_isRP)

RES->StringGrid1->Cells[8][\_nRows+1] = RES->StringGrid2->Cells[5][1];

if(\_isRP)

RES->StringGrid1->Cells[9][\_nRows+1] = RES->StringGrid2->Cells[8][1];

else

RES->StringGrid1->Cells[9][\_nRows+1] = RES->StringGrid2->Cells[6][1];

}

//âèäàëåííÿ îá'ºêòà

void CRealEstate :: vDeleteObject()

{

RES->Button5->Visible = false;

//int nLeft = RES->StringGrid1->Selection.Left;

int nTop = RES->StringGrid1->Selection.Top;

if(RES->StringGrid1->Cells[0][nTop] == "RP ")

--nCountRP;

else if(RES->StringGrid1->Cells[0][nTop] == "NRP" )

--nCountNRP;

vSetPrice(StrToInt(RES->StringGrid1->Cells[9][nTop]));

vShowPrice();

for(int i=0;i<RES->StringGrid1->ColCount;i++)

RES->StringGrid1->Cells[i][nTop] = "";

}

//âèâåäåííÿ çàãîëîâê³â

void CRealEstate :: vPrintHeaders()

{

RES->StringGrid1->Cells[0][0] = "Type";

RES->StringGrid1->Cells[1][0] = "Address ";

RES->StringGrid1->Cells[2][0] = "Owner";

RES->StringGrid1->Cells[3][0] = "Numbers of rooms";

RES->StringGrid1->Cells[4][0] = "Total area,m^2";

RES->StringGrid1->Cells[5][0] = "Kitchen area,m^2";

RES->StringGrid1->Cells[6][0] = "Rooms area,m^2";

RES->StringGrid1->Cells[7][0] = "Repair condition";

RES->StringGrid1->Cells[8][0] = "Purpose";

RES->StringGrid1->Cells[9][0] = "Price,$";

}

//âñòàíîâëåííÿ ñóìè ïðîäàíî¿ íåðóõîìîñò³

int CRealEstate :: s\_nPrice;

void CRealEstate :: vSetPrice(int \_nPrice)

{

s\_nPrice += \_nPrice;

}

//âèâ³ä çàãàëüíî¿ ñóìè ïðîäàíî¿ íåðóõîñò³

void CRealEstate :: vShowPrice()

{

RES->Edit2->Visible = true;

RES->Label3->Visible = true;

RES->Edit2->Text = IntToStr(s\_nPrice);

TStringList\* SL = new TStringList();

if(FileExists("Price.txt"))

SL->LoadFromFile("Price.txt");

SL->Clear();

SL->Add(RES->Edit2->Text.c\_str());

SL->SaveToFile("Price.txt");

delete SL;

}

Dfdsfsdf\AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

#include <vcl.h>

#include <fstream>

#include <iostream>

#include <cctype>

#include <map>

#include <string>

#include <sstream>

#pragma hdrstop

#pragma package(smart\_init)

#pragma resource "\*.dfm"

#include "File2.h"

#include "Unit1.h"

TRES \*RES;

\_\_fastcall TRES::TRES(TComponent\* Owner)

: TForm(Owner){}

using namespace std;

int nCheckSearch;

bool isRP;

CRealEstate MainObj;

CResidentialPremises \*objR;

CNonResidentialPremises \*objN;

//ç÷èòóâàííÿ, ³í³ö³àë³çàö³ÿ îá'ºêò³â, âèâåäåííÿ

void \_\_fastcall TRES::Start1Click(TObject \*Sender)

{

MainObj.vPrintHeaders();

MainObj.vReadFile();

vCreateObjects();

}

//î÷èùåííÿ StringGrid

void vClearStringGrid()

{

for(int i = 0;i < RES->StringGrid1->ColCount;i++)

for(int j = 1; j < RES->StringGrid1->RowCount;j++)

RES->StringGrid1->Cells[i][j] = "";

}

//ñòâîðåííÿ îá'ºêò³â

void vCreateObjects()

{

int nCounterRP = 0, nCounterNRP = 0;

objR = new CResidentialPremises[MainObj.nCountRP];

objN = new CNonResidentialPremises[MainObj.nCountNRP];

int nTmp = 0;

bool isFlag = true;//÷è ïóñòà ñòð³÷êà çóñòð³ëàñü ò³ëüêè ðàç

int i = 0;

while(i < (MainObj.nCountRP + MainObj.nCountNRP + nTmp)){

if(RES->StringGrid1->Cells[0][i+1] == "" && isFlag){

++nTmp;

isFlag = false;

}

else if(RES->StringGrid1->Cells[0][i+1]=="RP "){

objR[nCounterRP].vSetValueForRP(i);

++nCounterRP;

}

else if(RES->StringGrid1->Cells[0][i+1]=="NRP"){

objN[nCounterNRP].vSetValueForNRP(i);

++nCounterNRP;

}

++i;

}

RES->Edit3->Text = IntToStr(MainObj.nCountRP);

RES->Edit4->Text = IntToStr(MainObj.nCountNRP);

}

//ñîðòóâàííÿ çà ïðèçíà÷åííÿì

void \_\_fastcall TRES::PurposeazOnlyforNRP1Click(TObject \*Sender)

{

for(int i = 0; i < MainObj.nCountNRP - 1; ++i)

for(int j = 0; j < MainObj.nCountNRP - i - 1; ++j)

objN[j].vSortByPurpose(objN[j+1]);

vClearStringGrid();

for(int i = 0; i < MainObj.nCountNRP; ++i)

objN[i].vPrintNRPInForm(i);

}

//ñîðòóâàííÿ çà àäðåñîþ

void \_\_fastcall TRES::Addressaz1Click(TObject \*Sender)

{

for(int i = 0; i < MainObj.nCountRP - 1; ++i)

for(int j = 0; j < MainObj.nCountRP - i - 1; ++j)

objR[j].vSortByAddress(objR[j+1]);

for(int i = 0; i < MainObj.nCountNRP - 1; ++i)

for(int j = 0; j < MainObj.nCountNRP - i - 1; ++j)

objN[j].vSortByAddress(objN[j+1]);

vClearStringGrid();

int nCheck = 0, nCounterRP = 0, nCounterNRP = 0;

for(int i = 0;i < MainObj.nCountRP + MainObj.nCountNRP;++i)

{

if(nCounterRP > (MainObj.nCountRP - 1))

nCheck = 1;

else if( nCounterNRP > (MainObj.nCountNRP - 1))

nCheck = 2;

if((objR[nCounterRP].sGetAddressRP() < objN[nCounterNRP].sGetAddressNRP()

&& (nCounterRP < MainObj.nCountRP)) || nCheck == 2)

{

objR[nCounterRP].vPrintRPInForm(i);

++nCounterRP;

}

else if((objR[nCounterRP].sGetAddressRP() > objN[nCounterNRP].sGetAddressNRP()

&& (nCounterNRP < MainObj.nCountNRP)) || nCheck == 1)

{

objN[nCounterNRP].vPrintNRPInForm(i);

++nCounterNRP;

}

nCheck = 0;

}

}

//ñîðòóâàííÿ çà ê³ëüê³ñòþ ê³ìíàò

void \_\_fastcall TRES::Rooms1Click(TObject \*Sender)

{

for(int i = 0; i < MainObj.nCountRP - 1; ++i)

for(int j = 0; j < MainObj.nCountRP - i - 1; ++j)

objR[j].vSortByNumbersRooms(objR[j+1]);

for(int i = 0; i < MainObj.nCountNRP - 1; ++i)

for(int j = 0; j < MainObj.nCountNRP - i - 1; ++j)

objN[j].vSortByNumbersRooms(objN[j+1]);

vClearStringGrid();

int nCheck = 0, nCounterRP = 0, nCounterNRP = 0;

for(int i = 0;i < MainObj.nCountRP + MainObj.nCountNRP;++i)

{

if(nCounterRP > (MainObj.nCountRP - 1))

nCheck = 1;

else if( nCounterNRP > (MainObj.nCountNRP - 1))

nCheck = 2;

if((objR[nCounterRP].nGetNumbersRooms() < objN[nCounterNRP].nGetNumbersRooms()

&& (nCounterRP < MainObj.nCountRP)) || nCheck == 2)

{

objR[nCounterRP].vPrintRPInForm(i);

++nCounterRP;

}

else if((objR[nCounterRP].nGetNumbersRooms() > objN[nCounterNRP].nGetNumbersRooms()

&& (nCounterNRP < MainObj.nCountNRP)) || nCheck == 1)

{

objN[nCounterNRP].vPrintNRPInForm(i);

++nCounterNRP;

}

nCheck = 0;

}

}

//çíàõîäæåííÿ ìàêñèìàëüíî¿ ïëîù³ íåæèòëîâèõ ïðèì³ùåíü

void \_\_fastcall TRES::NRP1Click(TObject \*Sender)

{

int nCounterNRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountNRP - 1 ;++j)

if(objN[nCounterNRP].isDefinitionSquareMax(objN[j + 1]))

nCounterNRP = j + 1;

objN[nCounterNRP].vPrintNRPInForm(0);

}

//çíàõîäæåííÿ ì³í³ìàëüíî¿ ïëîù³ íåæèòëîâèõ ïðèì³ùåíü

void \_\_fastcall TRES::NRP2Click(TObject \*Sender)

{

int nCounterNRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountNRP - 1 ;++j)

if(objN[nCounterNRP].isDefinitionSquareMin(objN[j + 1]))

nCounterNRP = j + 1;

objN[nCounterNRP].vPrintNRPInForm(0);

}

//çíàõîäæåííÿ ìàêñèìàëüíî¿ ïëîù³ æèòëîâèõ ïðèì³ùåíü

void \_\_fastcall TRES::Rp1Click(TObject \*Sender)

{

int nCounterRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountRP - 1 ;++j)

if(objR[nCounterRP].isDefinitionSquareMax(objR[j + 1]))

nCounterRP = j + 1;

objR[nCounterRP].vPrintRPInForm(0);

}

//çíàõîäæåííÿ ì³í³ìàëüíî¿ ïëîù³ æèòëîâèõ ïðèì³ùåíü

void \_\_fastcall TRES::RP2Click(TObject \*Sender)

{

int nCounterRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountRP - 1 ;++j)

if(objR[nCounterRP].isDefinitionSquareMin(objR[j + 1]))

nCounterRP = j + 1;

objR[nCounterRP].vPrintRPInForm(0);

}

//çíàõîäæåííÿ ìàêñèìàëüíî¿ ïëîù³ óñ³õ ïðèì³ùåíü

void \_\_fastcall TRES::ALL1Click(TObject \*Sender)

{

int nCounterRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountRP - 1 ;++j)

if(objR[nCounterRP].isDefinitionSquareMax(objR[j + 1]))

nCounterRP = j + 1;

int nCounterNRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountNRP - 1 ;++j)

if(objN[nCounterNRP].isDefinitionSquareMax(objN[j + 1]))

nCounterNRP = j + 1;

if(objR[nCounterRP].nGetTotalArea() > objN[nCounterNRP].nGetTotalArea())

objR[nCounterRP].vPrintRPInForm(0);

else if(objR[nCounterRP].nGetTotalArea() < objN[nCounterNRP].nGetTotalArea())

objN[nCounterNRP].vPrintNRPInForm(0);

}

//çíàõîäæåííÿ ì³í³ìàëüíî¿ ïëîù³ óñ³õ ïðèì³ùåíü

void \_\_fastcall TRES::ALL2Click(TObject \*Sender)

{

int nCounterRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountRP - 1 ;++j)

if(objR[nCounterRP].isDefinitionSquareMin(objR[j + 1]))

nCounterRP = j + 1;

int nCounterNRP = 0;

vClearStringGrid();

for(int j = 0;j < MainObj.nCountNRP - 1 ;++j)

if(objN[nCounterNRP].isDefinitionSquareMin(objN[j + 1]))

nCounterNRP = j + 1;

if(objR[nCounterRP].nGetTotalArea() < objN[nCounterNRP].nGetTotalArea())

objR[nCounterRP].vPrintRPInForm(0);

else if(objR[nCounterRP].nGetTotalArea() > objN[nCounterNRP].nGetTotalArea())

objN[nCounterNRP].vPrintNRPInForm(0);

}

//âèá³ð ïîøóêó àäðåñè

void \_\_fastcall TRES::Address1Click(TObject \*Sender)

{

nCheckSearch = 1;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó âëàñíèêà

void \_\_fastcall TRES::Byparametr1Click(TObject \*Sender)

{

nCheckSearch = 2;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ê³ëüêîñò³ ê³ìíàò

void \_\_fastcall TRES::NumbersOf1Click(TObject \*Sender)

{

nCheckSearch = 3;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó çàãàëüíî¿ ïëîù³

void \_\_fastcall TRES::Totalarea1Click(TObject \*Sender)

{

nCheckSearch = 4;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ïëîù³ êóõí³

void \_\_fastcall TRES::Kitchenarea1Click(TObject \*Sender)

{

nCheckSearch = 5;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ïëîù³ ê³ìíàò

void \_\_fastcall TRES::Roomsarea1Click(TObject \*Sender)

{

nCheckSearch = 6;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ñòàíó ðåìîíòó

void \_\_fastcall TRES::Repaircondition1Click(TObject \*Sender)

{

nCheckSearch = 7;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ö³ëüîâîãî ïðèçíà÷åííÿ

void \_\_fastcall TRES::Purpose1Click(TObject \*Sender)

{

nCheckSearch = 8;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//âèá³ð ïîøóêó ö³íè

void \_\_fastcall TRES::Price1Click(TObject \*Sender)

{

nCheckSearch = 9;

RES->Button3->Visible = true;

RES->Edit1->Visible = true;

RES->Label4->Visible = true;

}

//ïîøóê çà çàäàíèìè ïàðàìåòðàìè

void \_\_fastcall TRES::Button3Click(TObject \*Sender)

{

vClearStringGrid();

int j = 0;

bool isCheckedSearch;

for(int i = 0;i < MainObj.nCountRP;++i){

isCheckedSearch = false;

switch(nCheckSearch){

case 1:

if(objR[i].isSearchAddress())

isCheckedSearch = true;

break;

case 2:

if(objR[i].isSearchOwner())

isCheckedSearch = true;

break;

case 3:

if(objR[i].isSearchNumbersOfRooms())

isCheckedSearch = true;

break;

case 4:

if(objR[i].isSearchTotalArea())

isCheckedSearch = true;

break;

case 5:

if(objR[i].isSearchKitchenArea())

isCheckedSearch = true;

break;

case 6:

if(objR[i].isSearchRoomsArea())

isCheckedSearch = true;

break;

case 7:

if(objR[i].isSearchRepairCondition())

isCheckedSearch = true;

break;

case 8:

break;

case 9:

if(objR[i].isSearchPrice())

isCheckedSearch = true;

break;

}

if(isCheckedSearch){

objR[i].vPrintRPInForm(j);

++j;

}

}

for(int i = 0;i < MainObj.nCountNRP;++i){

isCheckedSearch = false;

switch(nCheckSearch){

case 1:

if(objN[i].isSearchAddress())

isCheckedSearch = true;

break;

case 2:

if(objN[i].isSearchOwner())

isCheckedSearch = true;

break;

case 3:

if(objN[i].isSearchNumbersOfRooms())

isCheckedSearch = true;

break;

case 4:

if(objN[i].isSearchTotalArea())

isCheckedSearch = true;

break;

case 5:

break;

case 6:

break;

case 7:

break;

case 8:

if(objN[i].isSearchPurpose())

isCheckedSearch = true;

break;

case 9:

if(objN[i].isSearchPrice())

isCheckedSearch = true;

break;

}

if(isCheckedSearch){

objN[i].vPrintNRPInForm(j);

++j ;

}

}

}

//çíàõîäæåííÿ âëàñíèê³â, ÿê³ çàðåºñòðóâàëè á³ëüøå 2 ïðèì³ùåíü

void \_\_fastcall TRES::Whohave2RP1Click(TObject \*Sender)

{

//êîíòåéíåð äëÿ ïîøóêó âëàñíèêà,ÿêèé ìàº á³ëüøå äâîõ ïðèì³ùåíü

map <string,int> mOwnersMap;

map <string, int> :: iterator i;

string sOwners;

vClearStringGrid();

for(int i = 0;i < MainObj.nCountRP;++i)

sOwners +=objR[i].sGetOwner() + " ";

for(int i = 0;i < MainObj.nCountNRP;++i)

sOwners +=objN[i].sGetOwner() + " ";

istringstream ist(sOwners);

while (ist >> sOwners)

mOwnersMap[sOwners]++;

int j =0;

for (i = mOwnersMap.begin(); i != mOwnersMap.end(); ++i){

if(i->second >= 2){

for(int k = 0;k < MainObj.nCountRP;++k){

if(i->first == objR[k].sGetOwner()){

objR[k].vPrintRPInForm(j);

++j;

}

}

for(int k = 0;k < MainObj.nCountNRP;++k){

if(i->first == objN[k].sGetOwner()){

objN[k].vPrintNRPInForm(j);

++j;

}

}

}

}

}

//âèá³ð äîäàâàííÿ æèòëîâîãî ïðèì³ùåííÿ

void \_\_fastcall TRES::RP3Click(TObject \*Sender)

{

for(int i=0;i<RES->StringGrid2->ColCount;i++)

RES->StringGrid2->Cells[i][1] = "";

isRP = true;

StringGrid1->Visible = false;

StringGrid2->Visible = true;

Button4->Visible = true;

StringGrid2->ColCount = 9;

StringGrid2->ColWidths[7] = 93;

StringGrid2->Width = 709;

Button4->Left = 280;

StringGrid2->Cells[0][0] = "Type";

StringGrid2->Cells[1][0] = "Address ";

StringGrid2->Cells[2][0] = "Owner";

StringGrid2->Cells[3][0] = "Numbers of rooms";

StringGrid2->Cells[4][0] = "Total area,m^2";

StringGrid2->Cells[5][0] = "Kitchen area,m^2";

StringGrid2->Cells[6][0] = "Rooms area,m^2";

StringGrid2->Cells[7][0] = "Repair condition";

StringGrid2->Cells[8][0] = "Price,$";

}

//âèá³ð äîäàâàííÿ íåæèòëîâîãî ïðèì³ùåííÿ

void \_\_fastcall TRES::NRP3Click(TObject \*Sender)

{

isRP = false;

for(int i=0;i<RES->StringGrid2->ColCount;i++)

RES->StringGrid2->Cells[i][1] = "";

StringGrid1->Visible = false;

StringGrid2->Visible = true;

Button4->Visible = true;

StringGrid2->ColCount = 7;

StringGrid2->Width = 550;

Button4->Left = 190;

StringGrid2->Cells[0][0] = "Type";

StringGrid2->Cells[1][0] = "Address ";

StringGrid2->Cells[2][0] = "Owner";

StringGrid2->Cells[3][0] = "Numbers of rooms";

StringGrid2->Cells[4][0] = "Total area,m^2";

StringGrid2->Cells[5][0] = "Purpose";

StringGrid2->Cells[6][0] = "Price,$";

}

//äîäàâàííÿ ïðèì³ùåííÿ

void \_\_fastcall TRES::Button4Click(TObject \*Sender)

{

MainObj.vAddObjectToStringGrid(isRP);

delete []objR;

delete []objN;

if(isRP)

MainObj.nCountRP = ++MainObj.nCountRP;

else

MainObj.nCountNRP = ++MainObj.nCountNRP;

vCreateObjects();

}

//âèäàëåííÿ óñ³õ îá'ºêò³â

void \_\_fastcall TRES::AllElements1Click(TObject \*Sender)

{

delete [] objR;

delete [] objN;

vClearStringGrid();

RES->Edit3->Text = IntToStr(0);

RES->Edit4->Text = IntToStr(0);

}

//âèá³ð âèäàëåííÿ îáðàíîãî åëåìåíòà

void \_\_fastcall TRES::ChooseElement1Click(TObject \*Sender)

{

RES->Button5->Visible = true;

ShowMessage ("Choose!");

}

//âèäàëåííÿ îáðàíîãî ïðèì³ùåííÿ

void \_\_fastcall TRES::Button5Click(TObject \*Sender)

{

MainObj.vDeleteObject();

delete [] objR;

delete [] objN;

vCreateObjects();

vClearStringGrid();

int j = 0;

for(int i = 0; i < MainObj.nCountRP; ++i){

objR[i].vPrintRPInForm(j);

++j;

}

for(int i = 0; i < MainObj.nCountNRP; ++i){

objN[i].vPrintNRPInForm(j);

++j;

}

}

//çáååæåííÿ ó ôàéë

void \_\_fastcall TRES::Save1Click(TObject \*Sender)

{

TStringList \*List = new TStringList();

List->Clear();

string sStr;

for (int i = 1; i < StringGrid1->RowCount && i < (MainObj.nCountRP+MainObj.nCountNRP); ++i )

{

sStr = "";

for (int j = 0; j < StringGrid1->ColCount; j++)

{

sStr += "|" ;

sStr += (StringGrid1->Cells[j][i]).c\_str();

}

sStr += "|" ;

List->Add(sStr.c\_str());

}

List->SaveToFile("Result.txt");

}

//---------------------------------------------------------------------------