**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**К­УРСОВА** **РОБОТА**

з дисципліни «Об'єктно-орієнтоване програмування»

на тему: «Створення моделі продажу нерухомості»

Студент ПЗ-24

спеціальності 121

«Інженерія програмного забезпечення»

Вонс Ю. А.

Науковий керівник

доцент кафедри ПЗ, доцент, к.т.н.

Дяконюк Л. М.

Національна шкала: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Львів — 2017 рік

**ЗМІСТ**

1. Технічне завдання…………………………………………………………...3
2. Алгоритм розв’язку задачі …………………………………………………4
3. Діаграми UML ………………………………………………………………7
4. Код програми………………………………………………………………..9
5. Протокол роботи програми………………………………………………..29
6. Виняткові ситуації…………………………………………………………35
7. Інструкція користувача……………………………………………………38
   1. Компонент ПЗ…………………………………………………………..38
   2. Встановлення ПЗ……………………………………………………….38
   3. Налаштування ПЗ………………………………………………………38
   4. Базові функції ПЗ……………………………………………………….38
8. Висновок …………………………………………………………………...39
9. Список використаної літератури………………………………………….40

# ТЕХНІЧНЕ ЗАВДАННЯ

Варіант 5(90 балів)

Створити програму, що відтворює модель продажу нерухомості. Передбачається, що нерухомість є 2 типів: житлові та нежитлові приміщення. Всі приміщення мають загальну площу, кількість кімнат, адресу, власника, вартість. Житлові приміщення мають площі кухні, кімнат, стан ремонту. Нежитлові приміщення – ще додатково мають цільове призначення.

Потрібно створити наступні сервіси:

1. Наповнення магазину інформацією.
2. Здійснення пошуку приміщення за заданими параметрами.
3. Сортування приміщень за цільовим призначенням, адресою, кількістю кімнат.
4. Визначення приміщень, що мають максимальну та мінімальну площу.
5. Пошук власників, які зареєстрували більше 2 приміщень для продажу.
6. Долучення та вилучення приміщення.
7. Обчислення загальної суми проданої нерухомості.

# Алгоритм розв’язку задачі

Заповнити таблицю можна двома способами:

1. зчитати дані із файлу, при цьому файл повинен містить у папці проекту;
2. зчитати дані із клавіатури;

Алгоритм заповнення таблиці представлений на рис. 1.

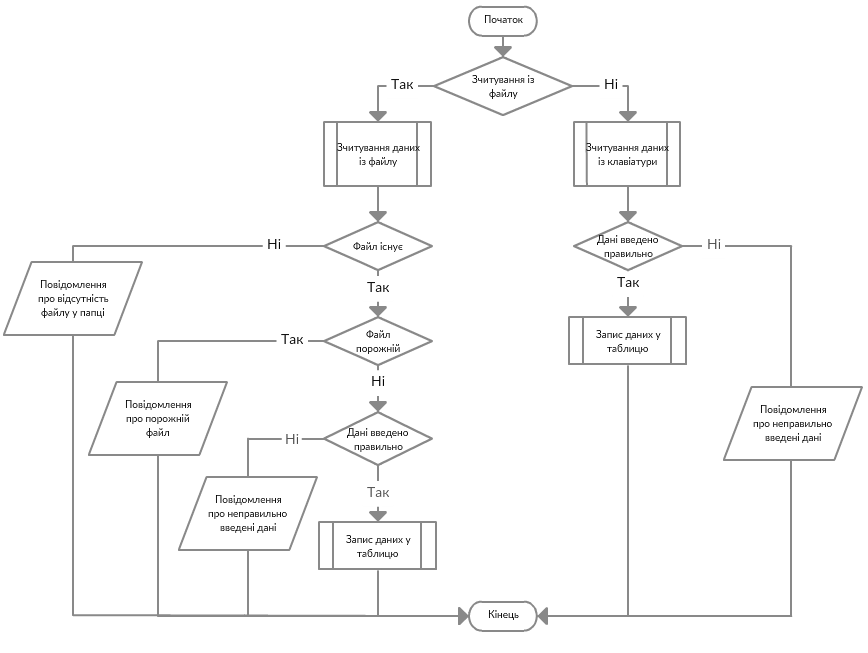


Рис 1. Блок-схема занесення даних

Для додавання даних необхідно вибрати тип нерухомості об’єкту та ввести дані. Алгоритм додавання даних у таблицю представлений на рис. 2.

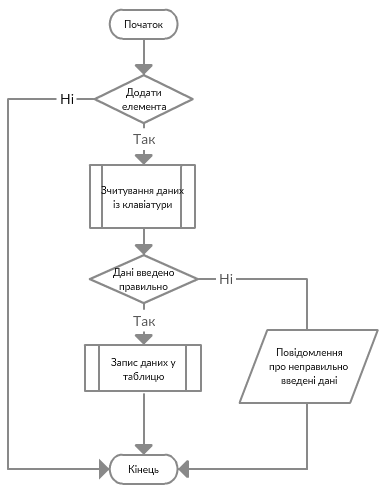


Рис 2. Блок-схема додавання даних

Для сортування даних необхідно вибрати метод сортування. Алгоритм сортування даних представлений на рис. 3.

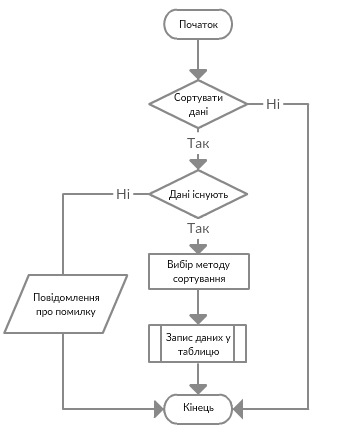


Рис 3. Блок-схема сортування даних

Для знаходження даних необхідно обрати тип параметру та ввести параметр. Алгоритм пошуку даних представлений на рис. 4.

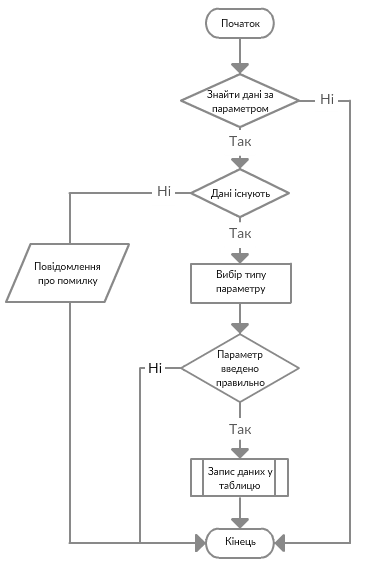


Рис 4. Блок-схема пошуку даних

# Діаграми UML

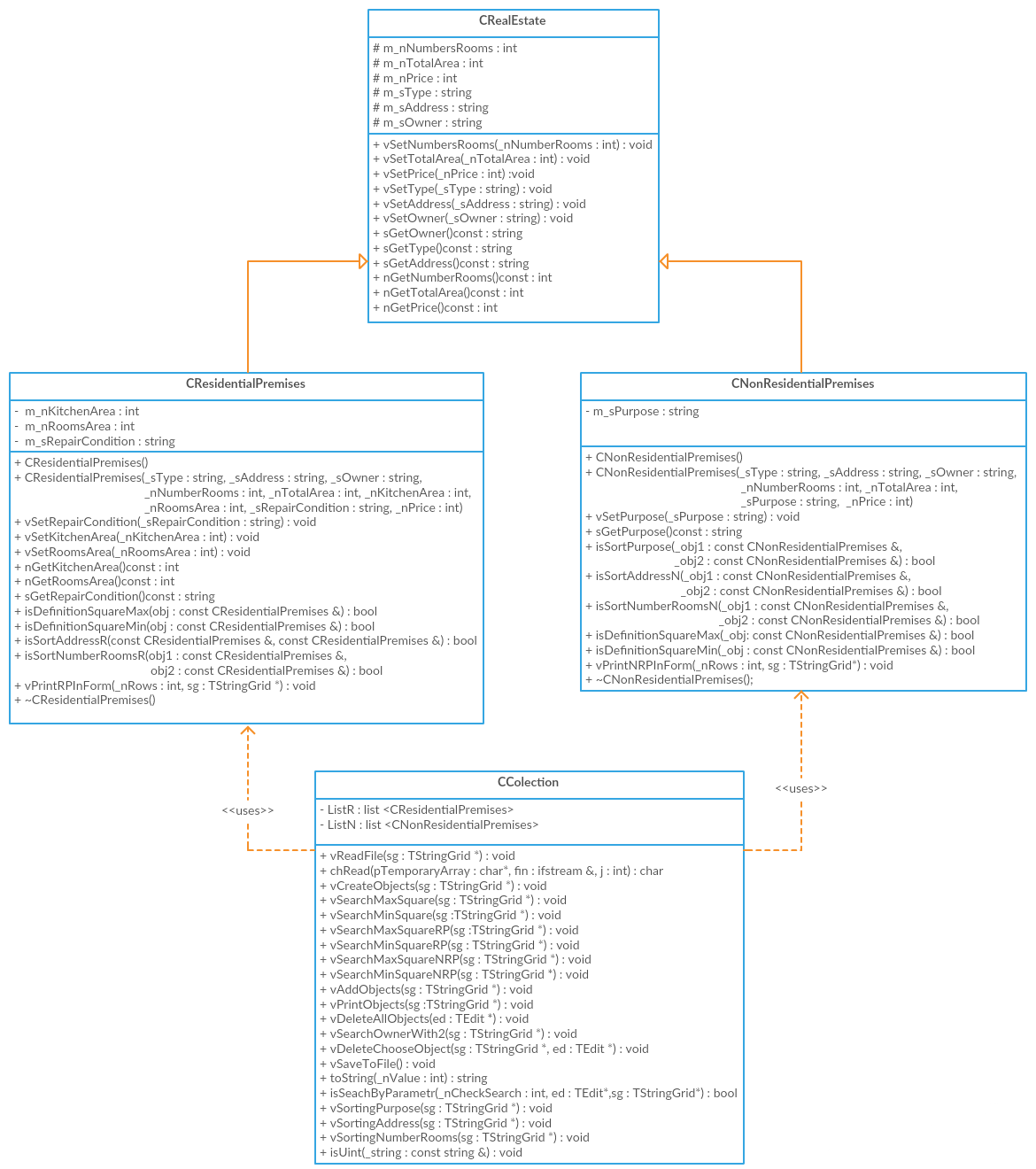


Рис 5. Діаграма класів

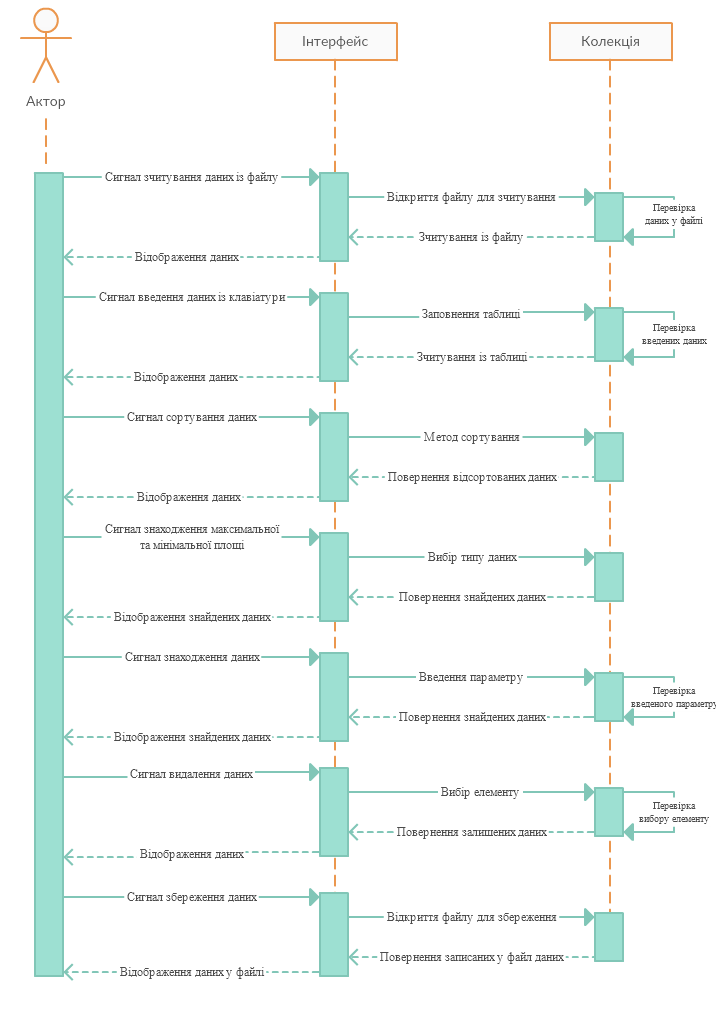


Рис 6. Діаграма послідовності

# Код програми

**RealEstate.h**

#ifndef HEADER

#define HEADER

class CRealEstate

{

protected:

int m\_nNumbersRooms;

int m\_nTotalArea;

int m\_nPrice;

string m\_sType;

string m\_sAddress;

string m\_sOwner;

public:

void vSetNumbersRooms(int \_nNumbersRooms);

void vSetTotalArea(int \_nTotalArea);

void vSetPrice(int \_nPrice);

void vSetType(string \_sType);

void vSetAddress(string \_sAddress);

void vSetOwner(string \_sOwner);

string sGetOwner()const;

string sGetType()const;

string sGetAddress()const;

int nGetNumberRooms()const;

int nGetTotalArea()const;

int nGetPrice()const;

};

class CResidentialPremises : public CRealEstate

{

private:

int m\_nKitchenArea;

int m\_nRoomsArea;

string m\_sRepairCondition;

public:

CResidentialPremises();

CResidentialPremises(string, string, string, int, int, int, int, string, int);

void vSetRepairCondition(string \_sRepairCondition);

void vSetKitchenArea(int \_nKitchenArea);

void vSetRoomsArea(int \_nRoomsArea);

int nGetKitchenArea()const;

int nGetRoomsArea()const;

string sGetRepairCondition()const;

bool isDefinitionSquareMax(const CResidentialPremises &obj);

bool isDefinitionSquareMin(const CResidentialPremises &obj);

friend bool isSortAddressR(const CResidentialPremises &obj1, const CResidentialPremises &obj2);

friend bool isSortNumberRoomsR(const CResidentialPremises &obj1, const CResidentialPremises &obj2);

void vPrintRPInForm(int \_nRows, TStringGrid \*sg);

~CResidentialPremises();

};

class CNonResidentialPremises : public CRealEstate

{

private:

string m\_sPurpose;

public:

CNonResidentialPremises();

CNonResidentialPremises(string, string, string, int, int, string, int);

void vSetPurpose(string \_sPurpose);

string sGetPurpose()const;

friend bool isSortPurpose(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2);

friend bool isSortAddressN(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2);

friend bool isSortNumberRoomsN(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2);

bool isDefinitionSquareMax(const CNonResidentialPremises &obj);

bool isDefinitionSquareMin(const CNonResidentialPremises &obj);

void vPrintNRPInForm(int \_nRows, TStringGrid \*sg);

~CNonResidentialPremises();

};

class CSummator

{

int sum ;

public:

int operator()(const CResidentialPremises &obj);

int operator()(const CNonResidentialPremises &obj);

int operator()(list <CResidentialPremises> l1, list <CNonResidentialPremises> l2);

};

class Exception1

{

protected:

int nNumberOfProblem;

string sProblem;

public:

virtual void vvDisplay() = 0;

};

class ReadFileException : public Exception1

{

int nProblemRow;

public:

ReadFileException(int , char \* , int );

void vvDisplay();

};

class AddInFormException : public Exception1

{

public:

AddInFormException(int , char \*);

void vvDisplay();

};

class SearchInFormException : public Exception1

{

public:

SearchInFormException(int , char \*);

void vvDisplay();

};

class DefinitionException : public Exception1

{

public:

DefinitionException(int , char \* );

void vvDisplay();

};

class OtherException : public Exception1

{

public:

OtherException(int , char \* );

void vvDisplay();

};

#endif

**CRealEstate.cpp**

//встановлення кількості кімнат

void CRealEstate :: vSetNumbersRooms(int \_nNumbersRooms)

{

m\_nNumbersRooms = \_nNumbersRooms;

}

//встановлення загальної площі

void CRealEstate :: vSetTotalArea(int \_nTotalArea)

{

m\_nTotalArea = \_nTotalArea;

}

//встановлення ціни

void CRealEstate :: vSetPrice(int \_nPrice)

{

m\_nPrice = \_nPrice;

}

//встановлення типу

void CRealEstate :: vSetType(string \_sType)

{

m\_sType = \_sType;

}

//встановлення адреси

void CRealEstate :: vSetAddress(string \_sAddress)

{

m\_sAddress = \_sAddress;

}

//встановлення імені власника

void CRealEstate :: vSetOwner(string \_sOwner)

{

m\_sOwner = \_sOwner;

}

//повернення типу

string CRealEstate :: sGetType()const

{

return m\_sType;

}

//повернення адреси

string CRealEstate :: sGetAddress()const

{

return m\_sAddress;

}

//повернення імені власника

string CRealEstate :: sGetOwner()const

{

return m\_sOwner;

}

//поверення кількості кімнат

int CRealEstate :: nGetNumberRooms()const

{

return m\_nNumbersRooms;

}

//поверення загальної площі

int CRealEstate :: nGetTotalArea()const

{

return m\_nTotalArea;

}

//поверення загальної ціни

int CRealEstate :: nGetPrice()const

{

return m\_nPrice;

}

//конструктор дочірнього класу без параметрів

CResidentialPremises :: CResidentialPremises()

{

m\_sType = "";

m\_sAddress = "";

m\_sOwner = "";

m\_nNumbersRooms = 0;

m\_nTotalArea = 0;

m\_nKitchenArea = 0;

m\_nRoomsArea = 0;

m\_sRepairCondition = "";

m\_nPrice = 0;

}

//конструктор дочірнього класу із параметрами

CResidentialPremises :: CResidentialPremises(string \_sType,string \_sAddress,string \_sOwner,int \_nNumberRooms,

int \_nTotalArea, int \_nKitchenArea, int \_nRoomsArea,string \_sRepairCondition,int \_nPrice)

{

m\_sType = \_sType;

m\_sAddress = \_sAddress;

m\_sOwner = \_sOwner;

m\_nNumbersRooms = \_nNumberRooms;

m\_nTotalArea = \_nTotalArea;

m\_nKitchenArea = \_nKitchenArea;

m\_nRoomsArea = \_nRoomsArea;

m\_sRepairCondition = \_sRepairCondition;

m\_nPrice = \_nPrice;

}

//встановлення цільового призначення

void CResidentialPremises :: vSetRepairCondition(string \_sRepairCondition)

{

m\_sRepairCondition = \_sRepairCondition;

}

//встановлення площі кухні

void CResidentialPremises :: vSetKitchenArea(int \_nKitchenArea)

{

m\_nKitchenArea = \_nKitchenArea ;

}

//встановлення площі кімнат

void CResidentialPremises :: vSetRoomsArea(int \_nRoomsArea)

{

m\_nRoomsArea = \_nRoomsArea ;

}

//повернення площі кухні

int CResidentialPremises :: nGetKitchenArea()const

{

return m\_nKitchenArea;

}

//повернення площі кімнат

int CResidentialPremises :: nGetRoomsArea()const

{

return m\_nRoomsArea;

}

//повернення цільового призначення

string CResidentialPremises :: sGetRepairCondition()const

{

return m\_sRepairCondition;

}

//перевірка знайденої максимальної площі

bool CResidentialPremises :: isDefinitionSquareMax(const CResidentialPremises &obj)

{

bool isReplace;

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

isReplace = true;

return isReplace;

}

//перевірка знайденої мінімальної площі

bool CResidentialPremises :: isDefinitionSquareMin(const CResidentialPremises &obj)

{

bool isReplace;

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

isReplace = true;

return isReplace;

}

//вивід об'єкта у StringGrid

void CResidentialPremises :: vPrintRPInForm(int \_nRows, TStringGrid \*sg)

{

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

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

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

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

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

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

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

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

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

}

//порівняння об'єктів за адресою

bool isSortAddressR(const CResidentialPremises &obj1,const CResidentialPremises &obj2)

{

return obj1.sGetAddress() < obj2.sGetAddress();

}

//порівняння об'єктів за кількістю кімнат

bool isSortNumberRoomsR(const CResidentialPremises &obj1,const CResidentialPremises &obj2)

{

return obj1.nGetNumberRooms() < obj2.nGetNumberRooms();

}

//деструктор дочірнього класу

CResidentialPremises :: ~CResidentialPremises()

{

}

//конструктор дочірнього класу без параметрів

CNonResidentialPremises :: CNonResidentialPremises()

{

m\_sType = "";

m\_sAddress = "";

m\_sOwner = "";

m\_nNumbersRooms = 0;

m\_nTotalArea = 0;

m\_sPurpose = "";

m\_nPrice = 0;

}

//конструктор дочірнього класу із параметрами

CNonResidentialPremises :: CNonResidentialPremises(string \_sType,string \_sAddress,string \_sOwner,int \_nNumberRooms,

int \_nTotalArea, string \_sPurpose,int \_nPrice)

{

m\_sType = \_sType;

m\_sAddress = \_sAddress;

m\_sOwner = \_sOwner;

m\_nNumbersRooms = \_nNumberRooms;

m\_nTotalArea = \_nTotalArea;

m\_sPurpose = \_sPurpose;

m\_nPrice = \_nPrice;

}

//встановлення цільового призначення

void CNonResidentialPremises :: vSetPurpose(string \_sPurpose)

{

m\_sPurpose = \_sPurpose;

}

//повернення цільового призначення

string CNonResidentialPremises :: sGetPurpose()const

{

return m\_sPurpose;

}

//перевірка знайденої максимальної площі

bool CNonResidentialPremises :: isDefinitionSquareMax(const CNonResidentialPremises &obj)

{

bool isReplace;

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

isReplace = true;

return isReplace;

}

//перевірка знайденої мінімальної площі

bool CNonResidentialPremises :: isDefinitionSquareMin(const CNonResidentialPremises &obj)

{

bool isReplace;

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

isReplace = true;

return isReplace;

}

//порівняння об'єктів за цільовим призначенням

bool isSortPurpose(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2)

{

return obj1.sGetPurpose() < obj2.sGetPurpose();

}

//порівняння об'єктів за адресою

bool isSortAddressN(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2)

{

return obj1.sGetAddress() < obj2.sGetAddress();

}

//порівняння об'єктів за кількістю кімнат

bool isSortNumberRoomsN(const CNonResidentialPremises &obj1,const CNonResidentialPremises &obj2)

{

return obj1.nGetNumberRooms() < obj2.nGetNumberRooms();

}

//вивід нежитлового об'єкта у StringGrid

void CNonResidentialPremises :: vPrintNRPInForm(int \_nRows, TStringGrid \*sg)

{

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

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

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

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

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

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

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

}

//деструктор дочірнього класу

CNonResidentialPremises :: ~CNonResidentialPremises()

{

}

//перевизначення оператора для житлових об'єктів

int CSummator :: operator()(const CResidentialPremises &obj)

{

sum = obj.nGetPrice();

return sum;

}

//перевизначення оператора для нежитлових об'єктів

int CSummator :: operator()(const CNonResidentialPremises &obj)

{

sum = obj.nGetPrice();

return sum;

}

//перевизначення оператора для усіх об'єктів

int CSummator :: operator()(list <CResidentialPremises> l1, list <CNonResidentialPremises> l2)

{

sum = 0;

for (list <CResidentialPremises> :: iterator it1 = l1.begin(); it1 != l1.end(); ++it1)

sum += it1->nGetPrice();

for (list <CNonResidentialPremises> :: iterator it2 = l2.begin(); it2 != l2.end(); ++it2)

sum += it2->nGetPrice();

return sum;

}

//констуктор класу обробки виключень зчитування із файлу

ReadFileException :: ReadFileException(int nNumberOfProblem, char \* sProblem, int nProblemRow)

{

this->nNumberOfProblem = nNumberOfProblem;

this->sProblem = sProblem;

this->nProblemRow = nProblemRow;

}

//вивід виключень для зчитування із файлу

void ReadFileException :: vvDisplay()

{

ShowMessage("Problem " + IntToStr(nNumberOfProblem) + " in " + IntToStr(nProblemRow)+

" row: " + sProblem.c\_str());

}

//констуктор класу обробки виключень додавання елементу

AddInFormException :: AddInFormException(int nNumberOfProblem, char \* sProblem)

{

this->nNumberOfProblem = nNumberOfProblem;

this->sProblem = sProblem;

}

//вивід виключень для додавання елементу

void AddInFormException :: vvDisplay()

{

ShowMessage("Problem " + IntToStr(nNumberOfProblem) + ": "+ sProblem.c\_str());

}

//констуктор класу обробки виключень пошуку елементу

SearchInFormException :: SearchInFormException(int nNumberOfProblem, char \* sProblem)

{

this->nNumberOfProblem = nNumberOfProblem;

this->sProblem = sProblem;

}

//вивід виключень для пошуку елементу

void SearchInFormException :: vvDisplay()

{

ShowMessage("Problem " + IntToStr(nNumberOfProblem) + ": "+ sProblem.c\_str());

}

//констуктор класу обробки виключень всіх інших функцій

OtherException :: OtherException(int nNumberOfProblem, char \* sProblem)

{

this->nNumberOfProblem = nNumberOfProblem;

this->sProblem = sProblem;

}

//вивід виключень для всіх інших

void OtherException :: vvDisplay()

{

ShowMessage("Problem " + IntToStr(nNumberOfProblem) +

" : " + sProblem.c\_str());

}

//констуктор класу обробки виключень всіх інших функцій

DefinitionException :: DefinitionException(int nNumberOfProblem, char \* sProblem)

{

this->nNumberOfProblem = nNumberOfProblem;

this->sProblem = sProblem;

}

//вивід виключень для всіх інших

void DefinitionException :: vvDisplay()

{

ShowMessage("Problem " + IntToStr(nNumberOfProblem) +

" : " + sProblem.c\_str());

}

**Collection.h**

#ifndef COLECTION\_H

#define COLECTION\_H

class CColection

{

private:

list <CResidentialPremises> ListR;

list <CNonResidentialPremises> ListN;

public:

void vReadFile(TStringGrid \*sg);

char chRead(char \*pTemporaryArray, ifstream &fin,int j);

void vCreateObjects(TStringGrid \*sg);

void vSearchMaxSquare(TStringGrid \*sg);

void vSearchMinSquare(TStringGrid \*sg);

void vSearchMaxSquareRP(TStringGrid \*sg);

void vSearchMinSquareRP(TStringGrid \*sg);

void vSearchMaxSquareNRP(TStringGrid \*sg);

void vSearchMinSquareNRP(TStringGrid \*sg);

void vAddObjects(TStringGrid \*sg);

void vPrintObjects(TStringGrid \*sg);

void vDeleteAllObjects(TEdit \*ed);

void vSearchOwnerWith2(TStringGrid \*sg);

void vDeleteChooseObject(TStringGrid \*sg, TEdit \*ed);

void vSaveToFile();

string toString(int \_nValue);

bool isSeachByParametr(int \_nCheckSearch, TEdit \*ed, TStringGrid \*sg);

void vSortingPurpose(TStringGrid \*sg);

void vSortingAddress(TStringGrid \*sg);

void vSortingNumberRooms(TStringGrid \*sg);

friend bool isUint(const string &);

};

#endif

**Collection.cpp**

//перевірка чи стрічка є числом

bool isUint(const string &s)

{

return s.find\_first\_not\_of("0123456789") == string::npos;

}

//приведення до типу string

string CColection :: toString(int \_nValue)

{

ostringstream oss;

oss << \_nValue;

return oss.str();

}

//зчитування слова із файлу

char CColection :: chRead(char \*pTemporaryArray, ifstream &fin,int j)

{

int i = 0;

char ch;

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

if(isdigit(ch))

throw ReadFileException(3, "You entered a digit instead word!", j + 1);

if (i < 10 && !isspace(ch))

{

pTemporaryArray[i] = ch;

++i;

pTemporaryArray[i] = '\0';

}

else

continue;

}

return \*pTemporaryArray;

}

//функція зчитування із файлу та записування у StringGrid

void CColection :: vReadFile(TStringGrid \*sg)

{

ifstream fin1("Read.txt");

if (!fin1)

throw OtherException(1, "No file in folder!");

else if(fin1.peek() == EOF)

throw OtherException(2, "File is empty!");

fin1.close();

ifstream fin("Read.txt");

char chInformation[200];

bool isInfo = true;

if(isInfo){

isInfo = false;

fin.getline(chInformation,200) ;

}

int c = 0, count , j = 0;

while(1){

if(sg->Cells[0][j+1] == "")

break;

++j;

}

bool isFlag;

char \*pTemporaryArray;

do{

if (fin.eof())

break;

c = 0;

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

//Type

pTemporaryArray = new char[10];

\*pTemporaryArray = chRead(pTemporaryArray, fin, j);

sg->Cells[c][j+1] = pTemporaryArray;

++c;

delete []pTemporaryArray;

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

isFlag = true;

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

isFlag = false;

//address

pTemporaryArray = new char[10];

\*pTemporaryArray = chRead(pTemporaryArray, fin, j);

sg->Cells[c][j+1] = pTemporaryArray;

++c;

delete []pTemporaryArray;

//Owner

pTemporaryArray = new char[10];

\*pTemporaryArray = chRead(pTemporaryArray, fin, j);

sg->Cells[c][j+1] = pTemporaryArray;

++c;

delete []pTemporaryArray;

//NumbersRooms

fin >> count;

if (fin.fail())

throw ReadFileException(4, "You don`t entered digit!",j+1);

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

if(IntToStr(count) < 0)

throw ReadFileException(5, "You entered a negative number!",j+1);

++c;

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

//TotalArea

fin >> count;

if (fin.fail())

throw ReadFileException(4, "You don`t entered digit!",j+1);

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

if(IntToStr(count) < 0)

throw ReadFileException(5, "You entered a negative number!",j+1);

++c;

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

if(isFlag){

//kitchenArea

fin >> count ;

if (fin.fail())

throw ReadFileException(4, "You don`t entered digit!", j+1);

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

if(IntToStr(count) < 0)

throw ReadFileException(5, "You entered a negative number!",j+1);

++c;

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

//Rooms area

fin >> count ;

if (fin.fail())

throw ReadFileException(4, "You don`t entered digit!",j+1);

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

if(IntToStr(count) < 0)

throw ReadFileException(5, "You entered a negative number!",j+1);

++c;

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

}

else{

++c;

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

++c;

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

}

if(isFlag){

//RepairCondition

pTemporaryArray = new char[10];

\*pTemporaryArray = chRead(pTemporaryArray, fin,j);

sg->Cells[c][j+1] = pTemporaryArray;

++c;

delete []pTemporaryArray;

}

else{

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

++c;

}

if(!isFlag){

//Purpose

pTemporaryArray = new char[10];

\*pTemporaryArray = chRead(pTemporaryArray, fin,j);

sg->Cells[c][j+1] = pTemporaryArray;

++c;

delete [] pTemporaryArray;

}

else{

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

++c;

}

fin >> count;

if (fin.fail())

throw ReadFileException(4, "You don`t entered digit!",j+1);

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

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

if(IntToStr(count) < 0)

throw ReadFileException(5, "You entered a negative number!",j+1);

++j;

fin.getline(chInformation,200) ;

}while(1);

fin.close();

}

//створення об'єктів

void CColection :: vCreateObjects(TStringGrid \*sg)

{

int i = 0;

while(1){

if(sg->Cells[0][i+1] == ""){

break;

}

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

CResidentialPremises \*objR = new CResidentialPremises;

objR->vSetType((sg->Cells[0][i+1]).c\_str());

objR->vSetAddress((sg->Cells[1][i+1]).c\_str());

objR->vSetOwner((sg->Cells[2][i+1]).c\_str());

objR->vSetNumbersRooms(StrToInt(sg->Cells[3][i+1]));

objR->vSetTotalArea(StrToInt(sg->Cells[4][i+1]));

objR->vSetKitchenArea(StrToInt(sg->Cells[5][i+1]));

objR->vSetRoomsArea(StrToInt(sg->Cells[6][i+1]));

objR->vSetRepairCondition((sg->Cells[7][i+1]).c\_str());

objR->vSetPrice(StrToInt(sg->Cells[9][i+1]));

ListR.push\_back(\*objR);

delete objR;

}

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

CNonResidentialPremises \*objN = new CNonResidentialPremises;

objN->vSetType((sg->Cells[0][i+1]).c\_str());

objN->vSetAddress((sg->Cells[1][i+1]).c\_str());

objN->vSetOwner((sg->Cells[2][i+1]).c\_str());

objN->vSetNumbersRooms(StrToInt(sg->Cells[3][i+1]));

objN->vSetTotalArea(StrToInt(sg->Cells[4][i+1]));

objN->vSetPurpose((sg->Cells[8][i+1]).c\_str());

objN->vSetPrice(StrToInt(sg->Cells[9][i+1]));

ListN.push\_back(\*objN);

delete objN;

}

++i;

}

}

//пошук найбільшої площі з усіх

void CColection :: vSearchMaxSquare(TStringGrid \*sg)

{

try

{

if(ListR.size() == 0 && ListN.size() == 0)

throw DefinitionException(6,"There are not elements!");

if(ListR.size() == 0)

throw DefinitionException(7,"There are not RP elements, please choose Definition only for NRP!");

if(ListN.size() == 0)

throw DefinitionException(8,"There are not NRP elements, please choose Definition only for RP!");

int k = 0;

CResidentialPremises \*obj1 = new CResidentialPremises;

\*obj1 = ListR.back();

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR)

if(itR->isDefinitionSquareMax(\*obj1))

\*obj1 = \*itR;

CNonResidentialPremises \*obj2 = new CNonResidentialPremises;

\*obj2 = ListN.back();

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin();itN != ListN.end();++itN)

if(itN->isDefinitionSquareMax(\*obj2))

\*obj2 = \*itN;

if(obj1->nGetTotalArea() >= obj2->nGetTotalArea()){

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj1->nGetTotalArea()){

itR->vPrintRPInForm(k, sg);

++k;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj1->nGetTotalArea()){

itN->vPrintNRPInForm(k, sg);

++k;

}

}

}

else{

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj2->nGetTotalArea()){

itR->vPrintRPInForm(k, sg);

++k;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj2->nGetTotalArea()){

itN->vPrintNRPInForm(k, sg);

++k;

}

}

}

delete obj1;

delete obj2;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//пошук найменшої площі з усіх

void CColection :: vSearchMinSquare(TStringGrid \*sg)

{

try

{

if(ListR.size() == 0 && ListN.size() ==0)

throw DefinitionException(6,"There are not elements!");

if(ListR.size() == 0)

throw DefinitionException(7,"There are not RP elements, please choose Definition only for NRP!");

if(ListN.size()== 0)

throw DefinitionException(8,"There are not NRP elements, please choose Definition only for RP!");

int k = 0;

CResidentialPremises \*obj1 = new CResidentialPremises;

\*obj1 = ListR.back();

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR)

if(itR->isDefinitionSquareMin(\*obj1))

\*obj1 = \*itR;

CNonResidentialPremises \*obj2 = new CNonResidentialPremises;

\*obj2 = ListN.back();

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin();itN != ListN.end();++itN)

if(itN->isDefinitionSquareMin(\*obj2))

\*obj2 = \*itN;

if(obj1->nGetTotalArea() >= obj2->nGetTotalArea()){

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj1->nGetTotalArea()){

itR->vPrintRPInForm(k, sg);

++k;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj1->nGetTotalArea()){

itN->vPrintNRPInForm(k, sg);

++k;

}

}

}

else{

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj2->nGetTotalArea()){

itR->vPrintRPInForm(k,sg);

++k;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj2->nGetTotalArea()){

itN->vPrintNRPInForm(k, sg);

++k;

}

}

}

delete obj1;

delete obj2;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//додавання елементу

void CColection :: vAddObjects(TStringGrid \*sg)

{

CResidentialPremises \*obj1;

CNonResidentialPremises \*obj2;

bool isRP;

if(sg->Cells[0][1] == "RP")

isRP = true;

else

isRP = false;

unsigned int i = 0;

if(sg->Cells[0][1] == "" )

throw AddInFormException(9,"You uncorrectly entered type!");

//Адреса

if(sg->Cells[1][1] == "" )

throw AddInFormException(16,"You uncorrectly entered address!");

string sTmpString3 = (sg->Cells[1][1]).c\_str();

i = 0;

while(i < sTmpString3.length()){

if(!isalpha(sTmpString3[i]))

throw AddInFormException(16,"You uncorrectly entered address!");

++i;

}

//Власник

if(sg->Cells[2][1] == "")

throw AddInFormException(17,"You uncorrectly entered owner!");

string sTmpString4 = (sg->Cells[2][1]).c\_str();

i = 0;

while(i < sTmpString4.length()){

if(!isalpha(sTmpString4[i]))

throw AddInFormException(17,"You uncorrectly entered owner!");

++i;

}

//Кількість кімнат

if(!isUint((sg->Cells[3][1]).c\_str()) || (sg->Cells[3][1] == ""))

throw AddInFormException(18,"You don`t entered a number in Number rooms!");

//Загальна площа

if(!isUint((sg->Cells[4][1]).c\_str()) || (sg->Cells[4][1] == ""))

throw AddInFormException(19,"You don`t entered a number in Total area!");

if(isRP){

//Тип

obj1 = new CResidentialPremises;

if(sg->Cells[0][1] != "RP")

throw AddInFormException(9,"You don`t entered a right type!");

//Площа кухні

if(!isUint((sg->Cells[5][1]).c\_str()) || (sg->Cells[5][1] == ""))

throw AddInFormException(10,"You don`t entered a number in Kitchen area!");

//Площа кімнат

if(!isUint((sg->Cells[6][1]).c\_str()) || (sg->Cells[6][1] == ""))

throw AddInFormException(11,"You don`t entered a number in Rooms area!");

//Ціна

if(!isUint((sg->Cells[8][1]).c\_str()) || (sg->Cells[8][1] == ""))

throw AddInFormException(13,"You don`t entered a number in Price!");

//чи сума площ не більша за загальну

if(StrToInt(sg->Cells[5][1]) < 0 || StrToInt(sg->Cells[6][1]) < 0 || StrToInt(sg->Cells[8][1]) < 0 ||

StrToInt(sg->Cells[3][1]) < 0 || StrToInt(sg->Cells[4][1]) < 0)

throw AddInFormException(5,"You entered a negative number!");

if((StrToInt(sg->Cells[6][1]) + StrToInt(sg->Cells[5][1]) )

> StrToInt(sg->Cells[4][1]))

throw AddInFormException(12,"Square of Kitchen and Rooms are bigger than Total area!");

//Стан ремонту

if(sg->Cells[7][1] == "")

throw AddInFormException(14,"You uncorrectly entered repair condition!");

string sTmpString1 = (sg->Cells[7][1]).c\_str();

i = 0;

while(i < sTmpString1.length()){

if(!isalpha(sTmpString1[i]) )

throw AddInFormException(14,"You uncorrectly entered repair condition!");

++i;

}

}

else if(!isRP){

//Тип

obj2 = new CNonResidentialPremises;

if(sg->Cells[0][1] != "NRP")

throw AddInFormException(9,"You don`t entered a right type!");

//Ціна

if(!isUint((sg->Cells[6][1]).c\_str()) || (sg->Cells[6][1] == ""))

throw AddInFormException(13,"You don`t entered a number in Price!");

if(StrToInt(sg->Cells[6][1]) < 0 || StrToInt(sg->Cells[3][1]) < 0 || StrToInt(sg->Cells[4][1]) < 0)

throw AddInFormException(5,"You entered a negative number!");

//Цільове призначення

if(sg->Cells[5][1] == "")

throw AddInFormException(15,"You uncorrectly entered purpose!");

string sTmpString2 = (sg->Cells[5][1]).c\_str();

i = 0;

while(i < sTmpString2.length()){

if(!isalpha(sTmpString2[i]))

throw AddInFormException(15,"You uncorrectly entered purpose!");

++i;

}

}

if(isRP){

obj1->vSetType((sg->Cells[0][1]).c\_str());

obj1->vSetAddress((sg->Cells[1][1]).c\_str());

obj1->vSetOwner((sg->Cells[2][1]).c\_str());

obj1->vSetNumbersRooms(StrToInt(sg->Cells[3][1]));

obj1->vSetTotalArea(StrToInt(sg->Cells[4][1]));

obj1->vSetKitchenArea(StrToInt(sg->Cells[5][1]));

obj1->vSetRoomsArea(StrToInt(sg->Cells[6][1]));

obj1->vSetRepairCondition((sg->Cells[7][1]).c\_str());

obj1->vSetPrice(StrToInt(sg->Cells[8][1]));

ListR.push\_back(\*obj1);

delete obj1;

}

else{

obj2->vSetType((sg->Cells[0][1]).c\_str());

obj2->vSetAddress((sg->Cells[1][1]).c\_str());

obj2->vSetOwner((sg->Cells[2][1]).c\_str());

obj2->vSetNumbersRooms(StrToInt(sg->Cells[3][1]));

obj2->vSetTotalArea(StrToInt(sg->Cells[4][1]));

obj2->vSetPurpose((sg->Cells[5][1]).c\_str());

obj2->vSetPrice(StrToInt(sg->Cells[6][1]));

ListN.push\_back(\*obj2);

delete obj2;

}

}

//вивід об'єктів

void CColection :: vPrintObjects(TStringGrid \*sg)

{

int i = 0;

for(list <CResidentialPremises> :: iterator itR = ListR.begin();itR != ListR.end();++itR, ++i)

itR->vPrintRPInForm(i, sg);

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin();itN !=ListN.end();++itN,++i)

itN->vPrintNRPInForm(i, sg);

}

//видалення усіх елементів

void CColection :: vDeleteAllObjects(TEdit \*ed)

{

ed->Text = IntToStr(CSummator()(ListR,ListN));

ListR.clear();

ListN.clear();

}

//пошук максимальноої площі RP

void CColection :: vSearchMaxSquareRP(TStringGrid \*sg)

{

try

{

if(ListR.size()== 0)

throw DefinitionException(7,"There are not RP elements!");

int k = 0;

CResidentialPremises \*obj1 = new CResidentialPremises;

\*obj1 = ListR.back();

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR)

if(itR->isDefinitionSquareMax(\*obj1))

\*obj1 = \*itR;

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj1->nGetTotalArea()){

itR->vPrintRPInForm(k, sg);

++k;

}

}

delete obj1;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//пошук мінімальної площі RP

void CColection :: vSearchMinSquareRP(TStringGrid \*sg)

{

try

{

if(ListR.size() ==0)

throw DefinitionException(7,"There are not RP elements!");

int k = 0;

CResidentialPremises \*obj1 = new CResidentialPremises;

\*obj1 = ListR.back();

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR)

if(itR->isDefinitionSquareMin(\*obj1))

\*obj1 = \*itR;

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(itR->nGetTotalArea() == obj1->nGetTotalArea()){

itR->vPrintRPInForm(k,sg);

++k;

}

}

delete obj1;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//пошук максимальної площі NRP

void CColection :: vSearchMaxSquareNRP(TStringGrid \*sg)

{

try

{

if(ListN.size() == 0)

throw DefinitionException(8,"There are not NRP elements!");

int k = 0;

CNonResidentialPremises \*obj1 = new CNonResidentialPremises;

\*obj1 = ListN.back();

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN)

if(itN->isDefinitionSquareMax(\*obj1))

\*obj1 = \*itN;

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj1->nGetTotalArea()){

itN->vPrintNRPInForm(k,sg);

++k;

}

}

delete obj1;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//пошук мінімальна площі NRP

void CColection :: vSearchMinSquareNRP(TStringGrid \*sg)

{

try

{

if(ListN.size() == 0)

throw DefinitionException(8,"There are not NRP elements!");

int k = 0;

CNonResidentialPremises \*obj1 = new CNonResidentialPremises;

\*obj1 = ListN.back();

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN)

if(itN->isDefinitionSquareMin(\*obj1))

\*obj1 = \*itN;

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(itN->nGetTotalArea() == obj1->nGetTotalArea()){

itN->vPrintNRPInForm(k,sg);

++k;

}

}

delete obj1;

}

catch(DefinitionException exp)

{

exp.vvDisplay();

}

}

//пошук власників, які мають більше двох нерухомостей

void CColection :: vSearchOwnerWith2(TStringGrid \*sg)

{

//контейнер для пошуку власника,який має більше двох приміщень

map <string,int> mOwnersMap;

map <string, int> :: iterator i;

string sOwners;

for(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR)

sOwners += itR->sGetOwner() + " ";

for(list <CNonResidentialPremises> :: iterator itN =ListN.begin(); itN != ListN.end();++itN)

sOwners += itN->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(list <CResidentialPremises> :: iterator itR = ListR.begin(); itR != ListR.end();++itR){

if(i->first == itR->sGetOwner()){

itR->vPrintRPInForm(j,sg);

++j;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

if(i->first == itN->sGetOwner())

{

itN->vPrintNRPInForm(j,sg);

++j;

}

}

}

}

}

//видалення обраного елемента

void CColection :: vDeleteChooseObject(TStringGrid \*sg, TEdit \*ed)

{

unsigned int nTop = sg->Selection.Top;

if(sg->Cells[0][nTop] == "")

throw OtherException(21,"You don`t choosed element!");

list <CResidentialPremises> :: iterator itR = ListR.begin();

list <CNonResidentialPremises> :: iterator itN = ListN.begin();

for(unsigned int i = 0; i < (ListR.size() + ListN.size())\*25;++i){

if(i == nTop && sg->Cells[0][i] == "RP"){

ed->Text = IntToStr(CSummator()(\*itR));

ListR.erase(itR);

break;

}

else if(i == nTop && sg->Cells[0][i] == "NRP"){

ed->Text = IntToStr(CSummator()(\*itN));

ListN.erase(itN);

break;

}

if(sg->Cells[0][i] == "RP")

++itR;

else if(sg->Cells[0][i] == "NRP")

++itN;

}

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

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

sg->Cells[i][j] = "";

vPrintObjects(sg);

}

//збереження у файл

void CColection :: vSaveToFile()

{

TStringList \*List = new TStringList();

List->Clear();

string sStr;

for(list <CResidentialPremises> :: iterator itR = ListR.begin();itR != ListR.end();++itR){

sStr = "";

sStr +="|";

sStr += itR->sGetType()+"|" + itR->sGetAddress() + "|" + itR->sGetOwner() +"|" +

toString(itR->nGetNumberRooms()) + "|" + toString(itR->nGetTotalArea()) + "|"

+toString(itR->nGetKitchenArea()) + "|" + toString(itR->nGetRoomsArea()) +

"|" + itR->sGetRepairCondition() + "|" + toString(itR->nGetPrice()) + "|" ;

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

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin(); itN != ListN.end();++itN){

sStr = "";

sStr +="|";

sStr += itN->sGetType()+ "|" + itN->sGetAddress() + "|" + itN->sGetOwner()

+ "|" + toString(itN->nGetNumberRooms()) + "|" + toString(itN->nGetTotalArea()) +

"|" +itN->sGetPurpose() + "|" + toString(itN->nGetPrice()) + "|" ;

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

}

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

}

//пошук за заданим параметром

bool CColection :: isSeachByParametr(int \_nCheckSearch ,TEdit \*ed, TStringGrid \*sg)

{

int j = 0;

if(ed->Text == "")

throw SearchInFormException(22,"You don`t entered a parametr!");

bool isCheckedSearch, isWasDeletingElement = false;

for(list <CResidentialPremises> :: iterator itR = ListR.begin();itR != ListR.end();++itR){

isCheckedSearch = false;

switch(\_nCheckSearch){

case 1:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(16,"You uncorrectly entered address!");

++i;

}

if(itR->sGetAddress() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 2:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(17,"You uncorrectly entered owner!");

++i;

}

if(itR->sGetOwner() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 3:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(18,"You don`t entered a number in number rooms!");

if(itR->nGetNumberRooms() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 4:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(19,"You don`t write a number in total area!");

if(itR->nGetTotalArea() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 5:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(10,"You don`t entered a number in kitchen square!");

if(itR->nGetKitchenArea() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 6:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(11,"You don`t entered a number in rooms area!");

if(itR->nGetRoomsArea() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 7:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(14,"You uncorrectly entered repair condition!");

++i;

}

if(itR->sGetRepairCondition() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 8:

break;

case 9:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(13,"You don`t entered a number in price!");

if(itR->nGetPrice() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

}

if(!isCheckedSearch && !isWasDeletingElement)

isWasDeletingElement = false;

else

isWasDeletingElement = true;

if(isCheckedSearch){

itR->vPrintRPInForm(j,sg);

++j;

}

}

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin();itN != ListN.end();++itN){

isCheckedSearch = false;

switch(\_nCheckSearch){

case 1:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(16,"You uncorrectly entered address!");

++i;

}

if(itN ->sGetAddress() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 2:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(17,"You uncorrectly entered owner!");

++i;

}

if(itN->sGetOwner() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 3:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(18,"You don`t entered a number in number rooms!");

if(itN->nGetNumberRooms() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 4:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(19,"You don`t entered a number in total area!");

if(itN->nGetTotalArea() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

case 8:{

string sTmpString = ed->Text.c\_str();

unsigned int i = 0;

while(i < sTmpString.length()){

if(!isalpha(sTmpString[i]))

throw SearchInFormException(15,"You uncorrectly entered purpose!");

++i;

}

if(itN->sGetPurpose() == ed->Text.c\_str())

isCheckedSearch = true;

break;

}

case 9:{

if(!isUint(ed->Text.c\_str()) || (ed->Text.c\_str() == ""))

throw SearchInFormException(13,"You don`t entered a number in price!");

if(itN->nGetPrice() == StrToInt(ed->Text))

isCheckedSearch = true;

break;

}

default:

break;

}

if(!isCheckedSearch && !isWasDeletingElement)

isWasDeletingElement = false;

else

isWasDeletingElement = true;

if(isCheckedSearch){

itN->vPrintNRPInForm(j,sg);

++j ;

}

}

if((!isCheckedSearch) && (isWasDeletingElement == false) )

isWasDeletingElement = false;

else

isWasDeletingElement = true;

return isWasDeletingElement;

}

//сортування за цільовим призначенням

void CColection :: vSortingPurpose(TStringGrid \*sg)

{

int i = 0;

ListN.sort(isSortPurpose);

for(list <CNonResidentialPremises> :: iterator itN = ListN.begin();itN != ListN.end();++itN, ++i)

itN->vPrintNRPInForm(i,sg);

}

//сортування за адресою

void CColection :: vSortingAddress(TStringGrid \*sg)

{

int i = 0;

ListR.sort(isSortAddressR);

ListN.sort(isSortAddressN);

list <CNonResidentialPremises> :: iterator itN = ListN.begin();

list <CResidentialPremises> :: iterator itR = ListR.begin();

while(1){

if(itN == ListN.end() && itR == ListR.end())

break;

if(itR->sGetAddress() <= itN->sGetAddress()){

itR->vPrintRPInForm(i,sg);

++itR;

++i;

if(itR == ListR.end()){

while(itN != ListN.end()){

itN->vPrintNRPInForm(i,sg);

++itN;

++i;

}

break;

}

}

else if(itN->sGetAddress() < itR->sGetAddress()){

itN->vPrintNRPInForm(i,sg);

++itN;

++i;

if(itN == ListN.end()){

while(itR != ListR.end()){

itR->vPrintRPInForm(i,sg);

++itR;

++i;

}

break;

}

}

}

}

//сортування за кількістю кімнат

void CColection :: vSortingNumberRooms(TStringGrid \*sg)

{

int i = 0;

ListR.sort(isSortNumberRoomsR);

ListN.sort(isSortNumberRoomsN);

list <CNonResidentialPremises> :: iterator itN = ListN.begin();

list <CResidentialPremises> :: iterator itR = ListR.begin();

while(1){

if(itN == ListN.end() && itR == ListR.end())

break;

if(itR->nGetNumberRooms() <= itN->nGetNumberRooms()){

itR->vPrintRPInForm(i,sg);

++itR;

++i;

if(itR == ListR.end()){

while(itN != ListN.end()){

itN->vPrintNRPInForm(i,sg);

++itN;

++i;

}

break;

}

}

else if(itN->nGetNumberRooms() < itR->nGetNumberRooms()){

itN->vPrintNRPInForm(i,sg);

++itN;

++i;

if(itN == ListN.end()){

while(itR != ListR.end()){

itR->vPrintRPInForm(i,sg);

++itR;

++i;

}

break;

}

}

}

}

# Протокол роботи програми

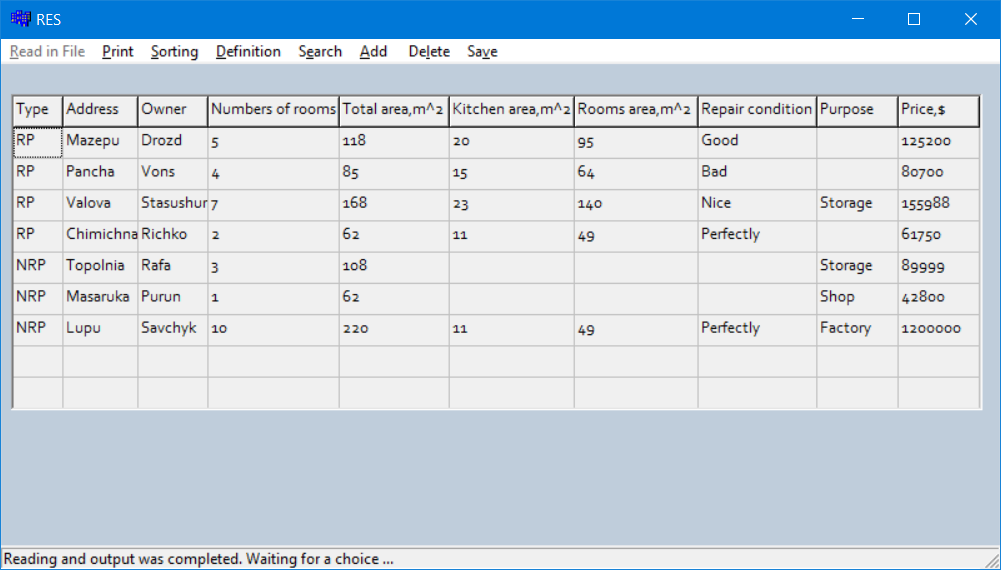


Рис 7. Вибір зчитування із файлу

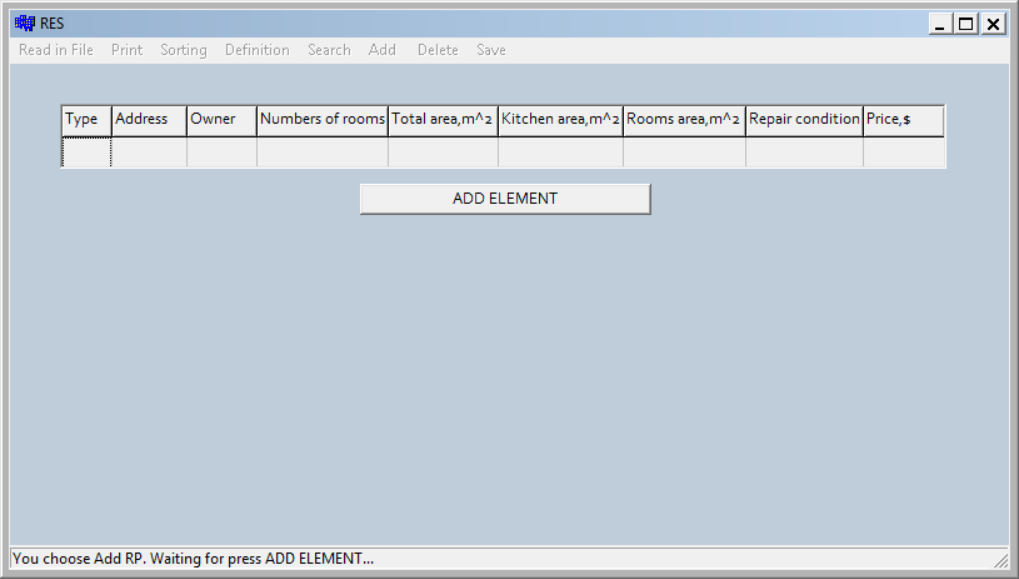


Рис 8. Вибір додавання елементу

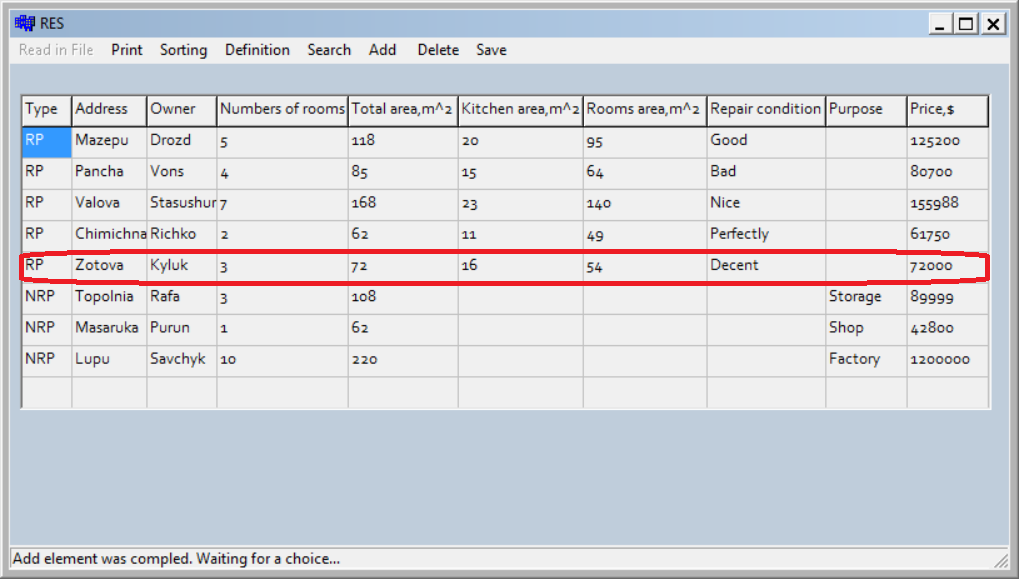


Рис 9. Додавання елементу до існуючого списку елементів

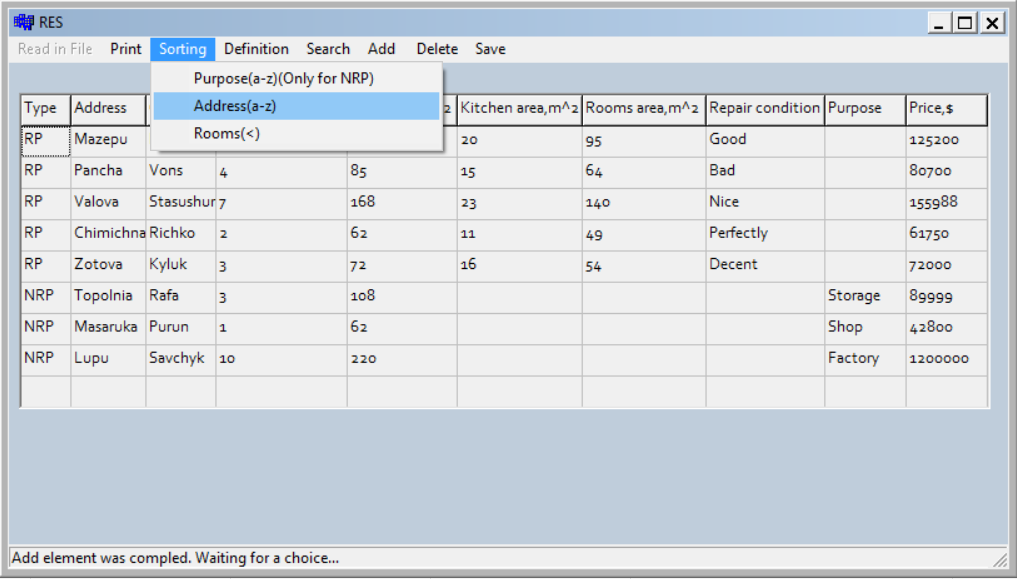


Рис 10. Меню сортування даних

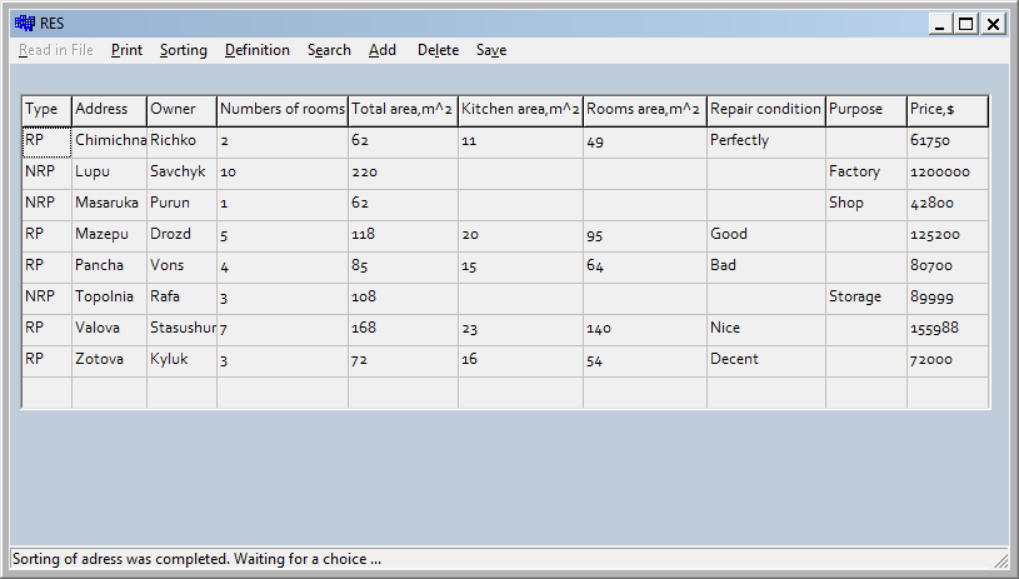


Рис 11. Посортовані дані за адресою

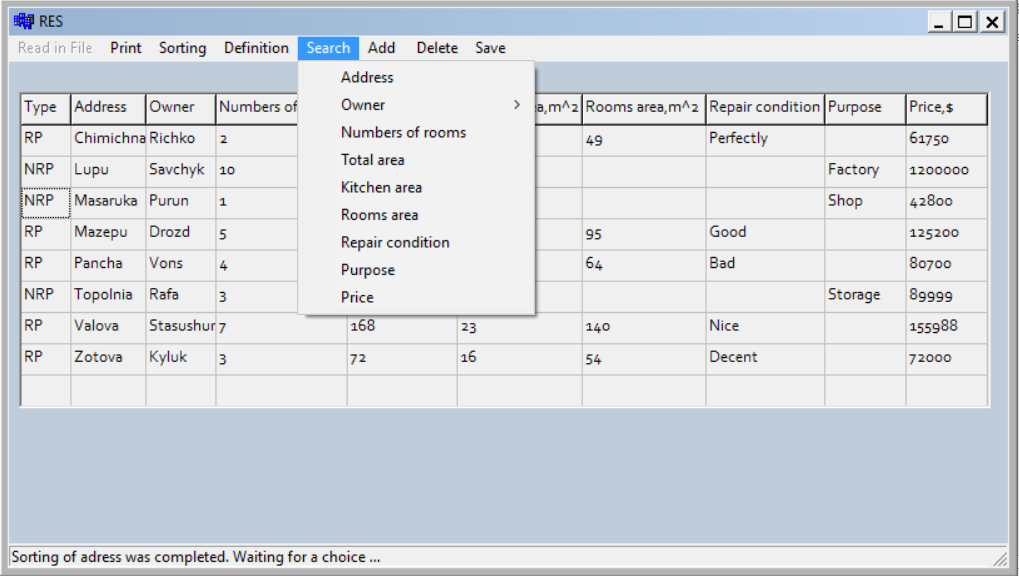


Рис 12. Меню пошуку

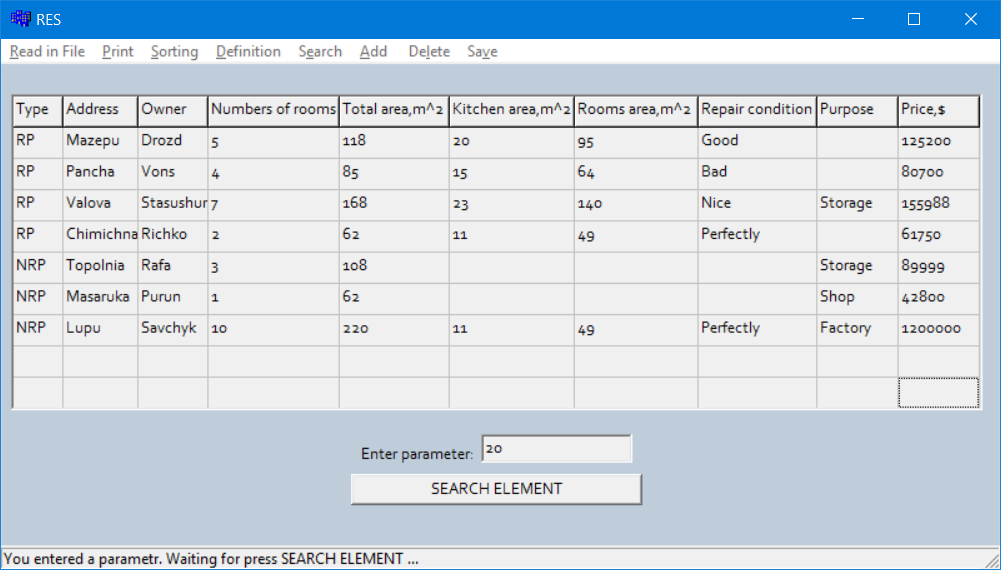


Рис 13. Відображення поля для введення параметру

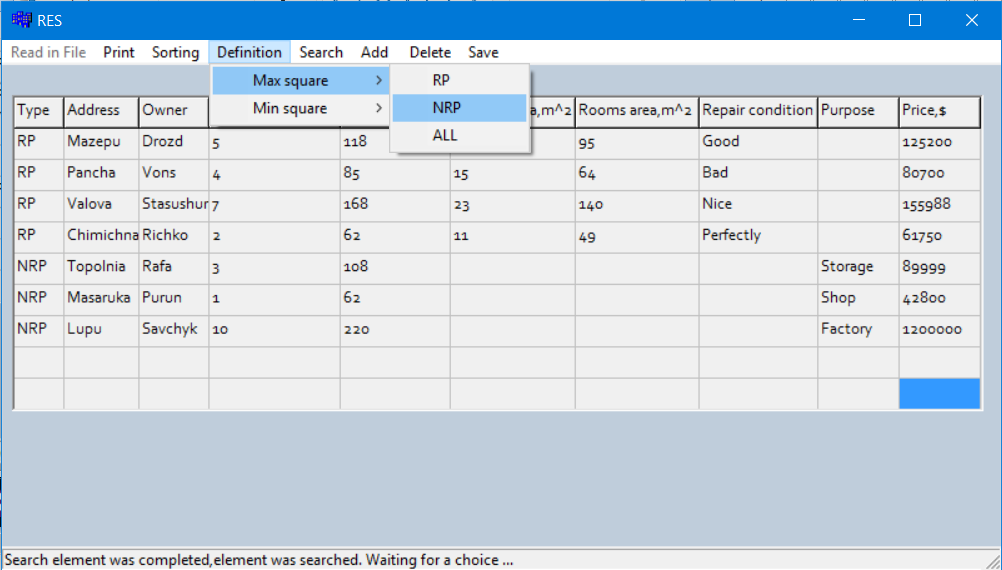


Рис 14. Меню знаходження максимальної та мінімальної площі

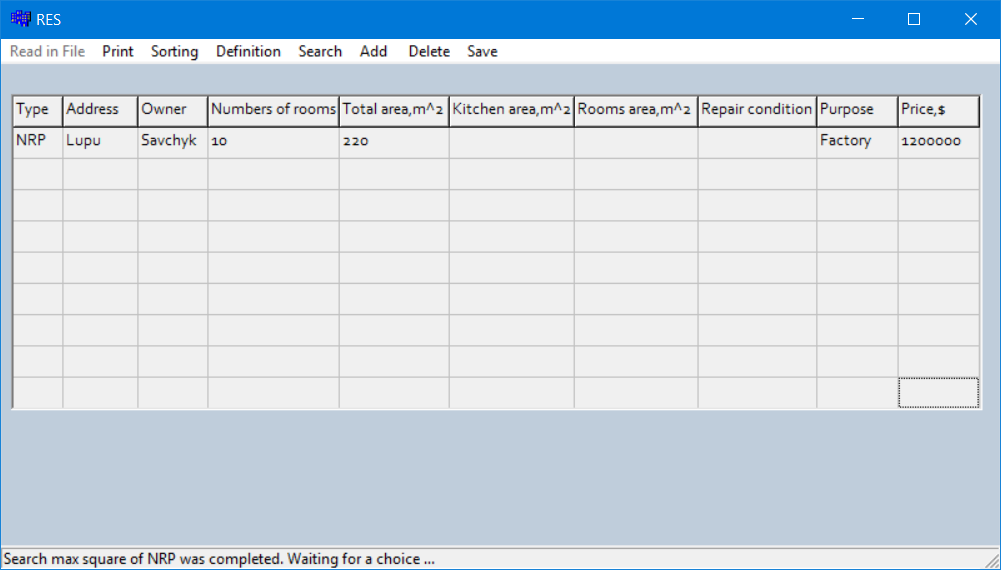


Рис 15. Знаходження нежитлового об’єкта із найбільшою площею

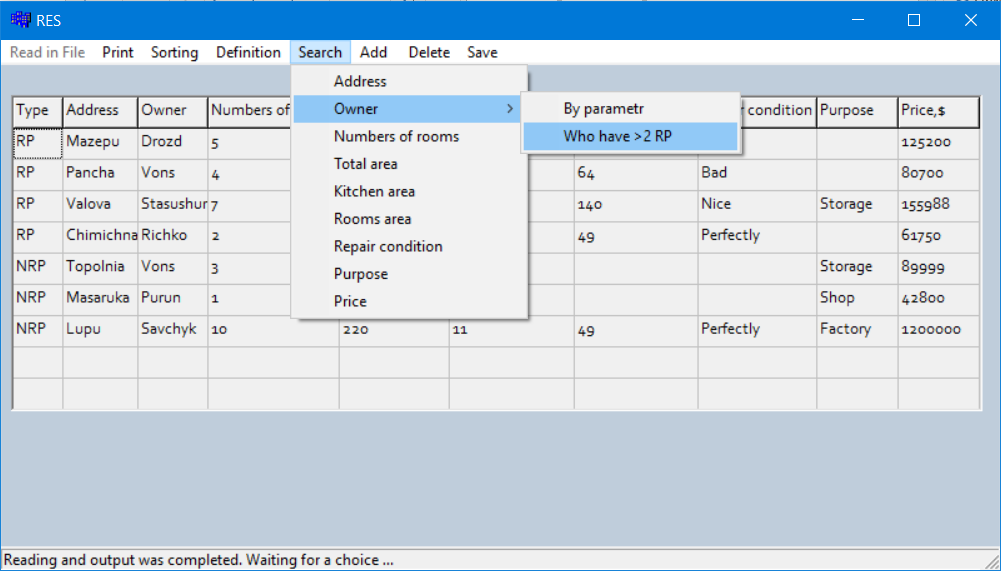


Рис 16. Вибір пошуку власника за умовою

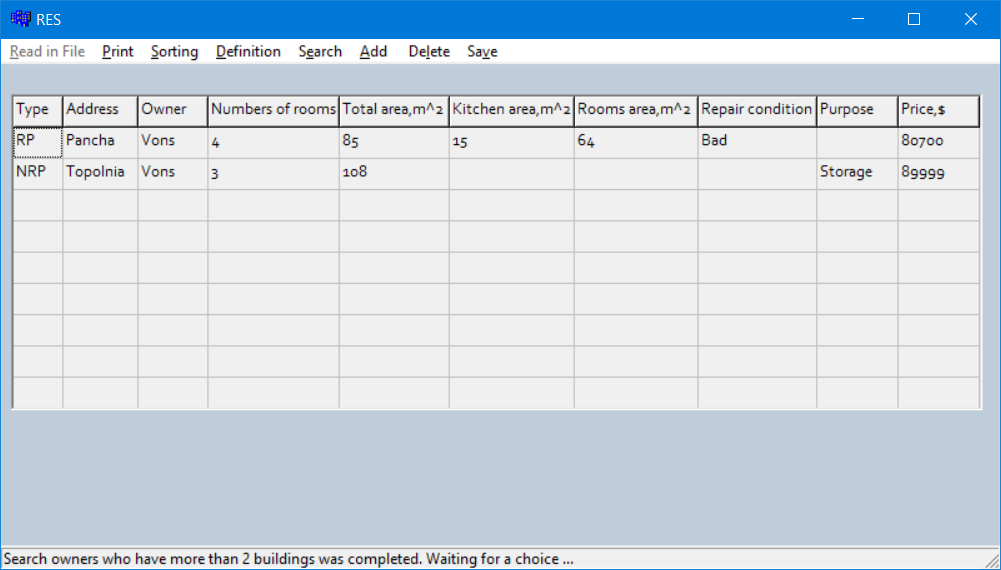


Рис 17. Знаходження власника, який має два або більше приміщень

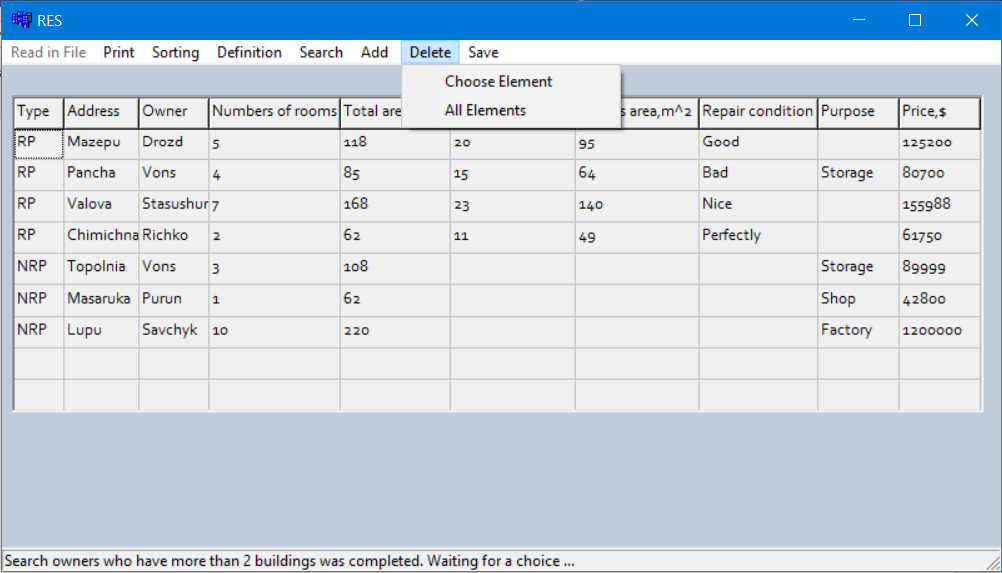


Рис 18. Меню видалення

# Виняткові ситуації

1. При зчитуванні із файлу:

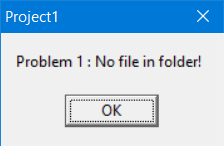


Рис 19. Не існує файлу у папці проекту

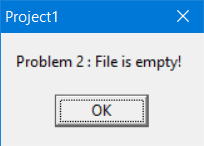


Рис 20. Файл пустий

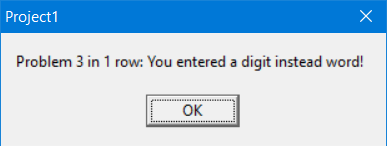


Рис 21. Введення числа замість слова

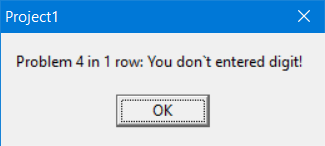


Рис 22. Введення не числа

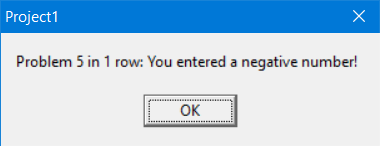


Рис 23. Введення від’ємного числа

1. При пошуку максимальної та мінімальної площі:

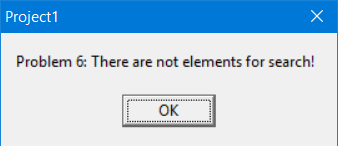


Рис 24. Список пустий

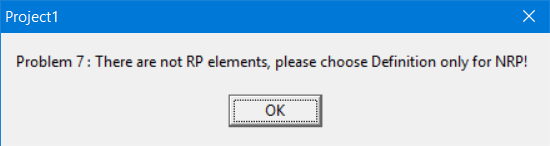


Рис 25. Не існує житлових об’єктів

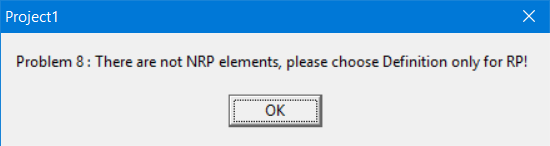


Рис 26. Не існує не житлових об’єктів

1. При додаванні даних у табличку та при пошуку даних виводиться помилка про неправильний ввід даних для прикладу:

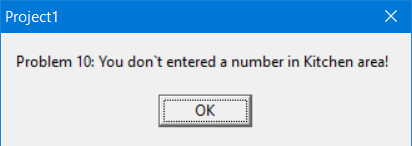


Рис 27. Неправильне введення площі кухні

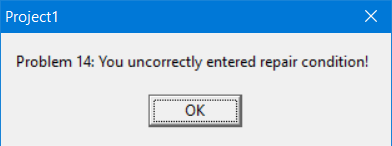


Рис 28. Неправильне введення стану ремонту

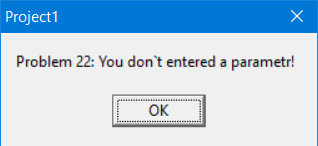


Рис 29. Не введення параметру для пошуку

1. При видаленні даних із таблички:

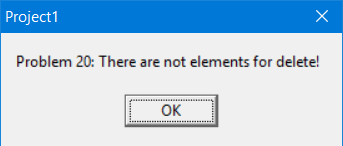


Рис 30. Список пустий

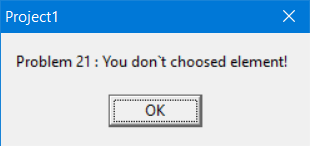


Рис 31. Не вибраний елемент

# Інструкція користувача

1. **Компоненти ПЗ**

Програма розроблена на мові програмування С++ у середовищі розробки C++ Builder 6 і може експлуатуватися під управлінням сімейства операційних систем Windows. Під час проектування підсистеми відбувалося поєднання об’єктно-орієнтованого підходу до програмування з процедурно-орієнтованим.

Для коректної роботи програмної системи необхідна користувацька машина з процесором не менше 1.6 ГГц та оперативною пам’яттю не менше 1ГБ.

1. **Встановлення ПЗ**

Програма потребує встановлення C++ Builder та запуск за допомогою файла з розширенням .exe.

1. **Налаштування ПЗ**

Для коректної роботи програми необхідно завантажити у папку проекту файл із даними.

1. **Базові функції ПЗ**

* Зчитування даних із файлу;
* Сортування даних за різним параметром;
* Знаходження вказаного елементу із списку даних;
* Знаходження мінімальної та максимальної площі;
* Додавання нових даних;
* Видалення даних;
* Збереження даних у файл;

# Висновок

Виконуючи курсову роботу засобами об’єктно-орієнтованого програмування на мові С++, я удосконалив вміння опрацювання даних, що включає у себе: ввід-вивід даних, перевірка на правильність написання даних, пошук інформації. Також я удосконалив навички створення форм у програмі «C++ Builder», а саме: побудова таблиць, використання MainMenu, StatusBar, вивід повідомлень про помилки. В процесі розробки програми я використовував бібліотеку шаблонів (STL), а саме, послідовний контейнер «List» та асоціативний контейнер «Map», який я раніше не використовував у своїй практиці. Також, ключовим завданням у моїй курсовій роботі було створення множинного наслідування, що дало мені змогу створювати об’єкти зі спільними властивостями без повторного написання коду.

# Список використаної літератури

1. Страуструп Б. Мова програмування С++ / Бьерн Страуструп., 2004. – 369 с.
2. Мейерс С. Ефективна мова програмування с++ / Скотт Мейерс., 2005. – 256 с. – (3).
3. Основи програмування на мовах Сі та Cі++ для початківців [Електронний ресурс] – Режим доступу до ресурсу: http://cppstudio.com/uk/.
4. cplusplus [Електронний ресурс] – Режим доступу до ресурсу: http://www.cplusplus.com.