Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»

Відділення другої вищої та післядипломної освіти

**ЛАБОРАТОРНА РОБОТА №2**

(варіант 2-16)

з курсу «Програмування»

на тему «Динамічні структури даних»

Виконав: студентка 3-го курсу

групи ІС-зп61

Шуміліна У.О.

Прийняв: викл. Древаль М.М.

Захищено з оцінкою

« » 2016 р.

Київ – 2016

# Умова завдання

Створити однозв’язний список мешканців одного мікрорайону, який містить персональні дані про кожного та адресу. Персональні дані включають наступну інформацію: прізвище, ім’я, по батькові, рік народження, площа житла. Адреса: вулиця, номер будинку, номер квартири.

1) Реалізувати функції перегляду записаних даних, додавання нового елемента на задану позицію (INSERT), видалення елемента з заданої позиції (DELETE). Початковий вміст списку заповнити з клавіатури.

2) Побудувати нове двійкове дерево одного окремого будинку (уводиться з клавіатури), вивести його мешканців у порядку зростання віку. Оформити як окрему функцію.

# Алгоритм розв’язання завдання

1. Створення однозв’язного списку мешканців одного мікрорайону
   1. За допомогою файла-БД
   2. За допомогою функції INSERT для пустого списку
2. Функції редагування однозвязного списку INSERT і DELETE
3. Процедура виводу результату в dataGridView1.
4. Створення дерева з елементами з використанням року народження у якості ключа, що містять інформацію вибраного будинку
5. Виведення в dataGridView1 елементів дерева від найбільшого до найменшого.
6. Збереження файлу даних.

# Лістинг програми

#pragma once

#include <string>

#include <fstream>

#include <msclr\marshal\_cppstd.h>

namespace Project1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

struct li {

li \* \_next = NULL;

int \_no;

std::string \_surname;

std::string \_name;

std::string \_pobatkovi;

int \_year;

std::string \_area;

std::string \_street;

std::string \_build;

std::string \_flat;

};

struct tr {

tr \* \_left = NULL;

tr \* \_right = NULL;

tr \* \_parent = NULL;

int \_year = NULL;

li \* item = NULL;

};

li \* first;

li \* all;

int last;

tr \* root;

int bound;

/// <summary>

/// Summary for MainForm

/// </summary>

public ref class MainForm : public System::Windows::Forms::Form

{

public:

MainForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MainForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ close\_but;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::Button^ load\_but;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_no;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_surname;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_name;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_pobatkovi;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_year;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_area;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_street;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_buidingNo;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ \_flatNo;

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Button^ del\_but;

private: System::Windows::Forms::TextBox^ textBox\_insert;

private: System::Windows::Forms::TextBox^ textBox\_bui;

private: System::Windows::Forms::TextBox^ textBox\_street;

private: System::Windows::Forms::Button^ sort\_but;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::CheckBox^ isBound;

private: System::Windows::Forms::CheckBox^ isShift;

private: System::Windows::Forms::CheckBox^ overwrite;

private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;

private: System::Windows::Forms::Button^ save\_but;

protected:

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->close\_but = (gcnew System::Windows::Forms::Button());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->\_no = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_surname = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_name = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_pobatkovi = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_year = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_area = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_street = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_buidingNo = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->\_flatNo = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->load\_but = (gcnew System::Windows::Forms::Button());

this->button3 = (gcnew System::Windows::Forms::Button());

this->del\_but = (gcnew System::Windows::Forms::Button());

this->textBox\_insert = (gcnew System::Windows::Forms::TextBox());

this->textBox\_bui = (gcnew System::Windows::Forms::TextBox());

this->textBox\_street = (gcnew System::Windows::Forms::TextBox());

this->sort\_but = (gcnew System::Windows::Forms::Button());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->isBound = (gcnew System::Windows::Forms::CheckBox());

this->isShift = (gcnew System::Windows::Forms::CheckBox());

this->overwrite = (gcnew System::Windows::Forms::CheckBox());

this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());

this->save\_but = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->SuspendLayout();

//

// close\_but

//

this->close\_but->Location = System::Drawing::Point(788, 346);

this->close\_but->Name = L"close\_but";

this->close\_but->Size = System::Drawing::Size(75, 23);

this->close\_but->TabIndex = 0;

this->close\_but->Text = L"Close";

this->close\_but->UseVisualStyleBackColor = true;

this->close\_but->Click += gcnew System::EventHandler(this, &MainForm::close\_but\_Click);

//

// dataGridView1

//

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(9) {

this->\_no, this->\_surname,

this->\_name, this->\_pobatkovi, this->\_year, this->\_area, this->\_street, this->\_buidingNo, this->\_flatNo

});

this->dataGridView1->Location = System::Drawing::Point(12, 12);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->Size = System::Drawing::Size(851, 239);

this->dataGridView1->TabIndex = 1;

//

// \_no

//

this->\_no->HeaderText = L"No";

this->\_no->Name = L"\_no";

this->\_no->Width = 50;

//

// \_surname

//

this->\_surname->HeaderText = L"Surname";

this->\_surname->Name = L"\_surname";

//

// \_name

//

this->\_name->HeaderText = L"Name";

this->\_name->Name = L"\_name";

//

// \_pobatkovi

//

this->\_pobatkovi->HeaderText = L"Pobatkovi";

this->\_pobatkovi->Name = L"\_pobatkovi";

//

// \_year

//

this->\_year->HeaderText = L"Year";

this->\_year->Name = L"\_year";

//

// \_area

//

this->\_area->HeaderText = L"Area";

this->\_area->Name = L"\_area";

this->\_area->Width = 50;

//

// \_street

//

this->\_street->HeaderText = L"Street";

this->\_street->Name = L"\_street";

//

// \_buidingNo

//

this->\_buidingNo->HeaderText = L"BuidingNo";

this->\_buidingNo->Name = L"\_buidingNo";

//

// \_flatNo

//

this->\_flatNo->HeaderText = L"FlatNo";

this->\_flatNo->Name = L"\_flatNo";

this->\_flatNo->Width = 80;

//

// load\_but

//

this->load\_but->Location = System::Drawing::Point(103, 257);

this->load\_but->Name = L"load\_but";

this->load\_but->Size = System::Drawing::Size(75, 23);

this->load\_but->TabIndex = 2;

this->load\_but->Text = L"Load";

this->load\_but->UseVisualStyleBackColor = true;

this->load\_but->Click += gcnew System::EventHandler(this, &MainForm::load\_but\_Click);

//

// button3

//

this->button3->Location = System::Drawing::Point(579, 257);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(75, 23);

this->button3->TabIndex = 3;

this->button3->Text = L"Insert";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MainForm::insert\_but\_Click);

//

// del\_but

//

this->del\_but->Location = System::Drawing::Point(579, 286);

this->del\_but->Name = L"del\_but";

this->del\_but->Size = System::Drawing::Size(75, 23);

this->del\_but->TabIndex = 4;

this->del\_but->Text = L"Delete";

this->del\_but->UseVisualStyleBackColor = true;

this->del\_but->Click += gcnew System::EventHandler(this, &MainForm::del\_but\_Click);

//

// textBox\_insert

//

this->textBox\_insert->Location = System::Drawing::Point(484, 286);

this->textBox\_insert->Name = L"textBox\_insert";

this->textBox\_insert->Size = System::Drawing::Size(89, 20);

this->textBox\_insert->TabIndex = 5;

//

// textBox\_bui

//

this->textBox\_bui->Location = System::Drawing::Point(484, 312);

this->textBox\_bui->Name = L"textBox\_bui";

this->textBox\_bui->Size = System::Drawing::Size(89, 20);

this->textBox\_bui->TabIndex = 6;

this->textBox\_bui->Text = L"77";

//

// textBox\_street

//

this->textBox\_street->Location = System::Drawing::Point(384, 312);

this->textBox\_street->Name = L"textBox\_street";

this->textBox\_street->Size = System::Drawing::Size(94, 20);

this->textBox\_street->TabIndex = 7;

this->textBox\_street->Text = L"Перемоги";

//

// sort\_but

//

this->sort\_but->Location = System::Drawing::Point(579, 315);

this->sort\_but->Name = L"sort\_but";

this->sort\_but->Size = System::Drawing::Size(75, 23);

this->sort\_but->TabIndex = 8;

this->sort\_but->Text = L"Sort";

this->sort\_but->UseVisualStyleBackColor = true;

this->sort\_but->Click += gcnew System::EventHandler(this, &MainForm::sort\_but\_Click);

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"sr1.txt";

//

// isBound

//

this->isBound->AutoSize = true;

this->isBound->Checked = true;

this->isBound->CheckState = System::Windows::Forms::CheckState::Checked;

this->isBound->Location = System::Drawing::Point(323, 315);

this->isBound->Name = L"isBound";

this->isBound->Size = System::Drawing::Size(43, 17);

this->isBound->TabIndex = 9;

this->isBound->Text = L"flag";

this->isBound->UseVisualStyleBackColor = true;

//

// isShift

//

this->isShift->AutoSize = true;

this->isShift->Location = System::Drawing::Point(323, 261);

this->isShift->Name = L"isShift";

this->isShift->Size = System::Drawing::Size(43, 17);

this->isShift->TabIndex = 10;

this->isShift->Text = L"flag";

this->isShift->UseVisualStyleBackColor = true;

//

// overwrite

//

this->overwrite->AutoSize = true;

this->overwrite->Location = System::Drawing::Point(384, 261);

this->overwrite->Name = L"overwrite";

this->overwrite->Size = System::Drawing::Size(69, 17);

this->overwrite->TabIndex = 11;

this->overwrite->Text = L"overwrite";

this->overwrite->UseVisualStyleBackColor = true;

//

// saveFileDialog1

//

this->saveFileDialog1->FileName = L"sr2.txt";

//

// save\_but

//

this->save\_but->Location = System::Drawing::Point(103, 286);

this->save\_but->Name = L"save\_but";

this->save\_but->Size = System::Drawing::Size(75, 23);

this->save\_but->TabIndex = 12;

this->save\_but->Text = L"Save";

this->save\_but->UseVisualStyleBackColor = true;

this->save\_but->Click += gcnew System::EventHandler(this, &MainForm::save\_but\_Click);

//

// MainForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(875, 381);

this->Controls->Add(this->save\_but);

this->Controls->Add(this->overwrite);

this->Controls->Add(this->isShift);

this->Controls->Add(this->isBound);

this->Controls->Add(this->sort\_but);

this->Controls->Add(this->textBox\_street);

this->Controls->Add(this->textBox\_bui);

this->Controls->Add(this->textBox\_insert);

this->Controls->Add(this->del\_but);

this->Controls->Add(this->button3);

this->Controls->Add(this->load\_but);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->close\_but);

this->Name = L"MainForm";

this->Text = L"MainForm";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: Void close\_but\_Click(Object^ sender, EventArgs^ e);

private: Void load\_but\_Click(Object^ sender, EventArgs^ e);

private: Void insert\_but\_Click(Object^ sender, EventArgs^ e);

private: std::string getFromLastRowCellNo(int num);

private: int getIntFromLastRowCellNo(int num);

private: Void ClearAllAndPrint();

private: Void del\_but\_Click(Object^ sender, EventArgs^ e);

private: tr \* addElemToTree(tr \*\* cur\_root, li \* a, int year);

private: tr \* prevElemTree(tr \* cur\_root);

private: tr \* prevElemTree22(tr \* cur\_root);

private: tr \* maxElemTree(tr \* cur\_root);

private: Void sort\_but\_Click(Object^ sender, EventArgs^ e);

private: Void save\_but\_Click(Object^ sender, EventArgs^ e);

};

}

//==================== MainForm.cpp ====================================

#include "MainForm.h"

namespace Project1 {

[STAThread]

void Main(array<String^>^ args) {

Application::EnableVisualStyles;

Application::SetCompatibleTextRenderingDefault(false);

MainForm myform;

Application::Run(%myform);

}

Void MainForm::close\_but\_Click(Object^ sender, EventArgs^ e) {

Close();

}

// LISTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Void MainForm::load\_but\_Click(Object^ sender, EventArgs^ e) {

if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

std::ifstream infile;

infile.open(msclr::interop::marshal\_as<std::string>(openFileDialog1->FileName));

char\*temp;

temp = (char\*)malloc(100 \* sizeof(char));

char del = ';';

first = new li;

all = first;

if (infile.is\_open())

{

while (!infile.eof())

{

infile.getline(temp, 100 \* sizeof(char), del);

all->\_no = atoi(temp);

infile.getline(temp, 100 \* sizeof(char), del);

all->\_surname = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_name = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_pobatkovi = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_year = atoi(temp);

infile.getline(temp, 100 \* sizeof(char), del);

all->\_area = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_street = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_build = temp;

infile.getline(temp, 100 \* sizeof(char), del);

all->\_flat = temp;

all->\_next = new li;

all = all->\_next;

//break;

}

infile.close();

ClearAllAndPrint();

}

}

}

Void MainForm::save\_but\_Click(Object^ sender, EventArgs^ e) {

if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK) {

std::ofstream infile;

infile.open(msclr::interop::marshal\_as<std::string>(saveFileDialog1->FileName));

all = first;

do {

infile << all->\_no << ";" << all->\_surname << ";" << all->\_name << ";" << all->\_pobatkovi << ";" << all->\_year << ";" << all->\_area << ";" << all->\_street << ";" << all->\_build << ";" << all->\_flat << ";\n";

all = all->\_next;

if (all->\_next == NULL) break;

} while (all->\_next->\_next != NULL);

infile.close();

}

}

Void MainForm::insert\_but\_Click(Object^ sender, EventArgs^ e) {

li \* temp = new li;

temp->\_next = new li;

temp->\_no = getIntFromLastRowCellNo(0);

temp->\_surname = getFromLastRowCellNo(1);

temp->\_name = getFromLastRowCellNo(2);

temp->\_pobatkovi = getFromLastRowCellNo(3);

temp->\_year = getIntFromLastRowCellNo(4);

temp->\_area = getFromLastRowCellNo(5);

temp->\_street = getFromLastRowCellNo(6);

temp->\_build = getFromLastRowCellNo(7);

temp->\_flat = getFromLastRowCellNo(8);

all = first;

if (all == NULL) {

first = temp;

ClearAllAndPrint();

return;

}

while (all->\_next != NULL) {

if ((all->\_no < temp->\_no) && (all->\_next->\_no >= temp->\_no)) break;

all = all->\_next;

}

if (all->\_next == NULL) {

}

else if (temp->\_no == all->\_next->\_no) {

if (overwrite->Checked) {

temp->\_next = all->\_next->\_next;

}

else {

temp->\_next = all->\_next;

}

}

else {

temp->\_next = all->\_next;

}

all->\_next = temp;

if (isShift->Checked) {

all = first;

while (all->\_next != NULL) {

if (all->\_no == all->\_next->\_no) all->\_next->\_no++;

all = all->\_next;

}

}

ClearAllAndPrint();

}

std::string MainForm::getFromLastRowCellNo(int num) {

return msclr::interop::marshal\_as<std::string>(dataGridView1->Rows[last]->Cells[num]->Value->ToString());

}

int MainForm::getIntFromLastRowCellNo(int num) {

return Convert::ToInt32(dataGridView1->Rows[last]->Cells[num]->Value);

}

Void MainForm::ClearAllAndPrint() {

dataGridView1->Rows->Clear();

int i = 0;

all = first;

do {

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->Cells[0]->Value = Convert::ToString(all->\_no);

dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(all->\_surname.c\_str());

dataGridView1->Rows[i]->Cells[2]->Value = gcnew String(all->\_name.c\_str());

dataGridView1->Rows[i]->Cells[3]->Value = gcnew String(all->\_pobatkovi.c\_str());

dataGridView1->Rows[i]->Cells[4]->Value = Convert::ToString(all->\_year);

dataGridView1->Rows[i]->Cells[5]->Value = gcnew String(all->\_area.c\_str());

dataGridView1->Rows[i]->Cells[6]->Value = gcnew String(all->\_street.c\_str());

dataGridView1->Rows[i]->Cells[7]->Value = gcnew String(all->\_build.c\_str());

dataGridView1->Rows[i]->Cells[8]->Value = gcnew String(all->\_flat.c\_str());

all = all->\_next;

i++;

if (all->\_next == NULL) break;

} while (all->\_next->\_next != NULL);

last = dataGridView1->Rows->Count - 1;

}

Void MainForm::del\_but\_Click(Object^ sender, EventArgs^ e) {

int del = Convert::ToInt32(textBox\_insert->Text);

li \* prev;

int i = 0;

if (del < 1) {

}

else if (del == 1) {

first = first->\_next;

}

else {

all = first;

while (all->\_next != NULL) {

if (all->\_no == del) {

prev->\_next = all->\_next;

break;

}

prev = all;

all = all->\_next;

i++;

}

}

ClearAllAndPrint();

}

// TREES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

tr \* MainForm::addElemToTree(tr \*\* cur\_root, li \* a, int year) {

if ((\*cur\_root) == NULL) {

(\*cur\_root) = new tr;

(\*cur\_root)->item = a;

(\*cur\_root)->\_year = year;

}

else if (year < (\*cur\_root)->\_year) {

(\*cur\_root)->\_left = addElemToTree(&(\*cur\_root)->\_left, a, year);

(\*cur\_root)->\_left->\_parent = (\*cur\_root);

}

else if (year >(\*cur\_root)->\_year) {

(\*cur\_root)->\_right = addElemToTree(&(\*cur\_root)->\_right, a, year);

(\*cur\_root)->\_right->\_parent = (\*cur\_root);

}

return (\*cur\_root);

}

tr \* MainForm::prevElemTree(tr \* cur\_root) {

if (cur\_root->\_left != NULL)

return maxElemTree(cur\_root->\_left);

tr \* current = root;

tr \* pre = NULL;

while (current != NULL) {

if (current->\_year < cur\_root->\_year) {

pre = current;

current = current->\_right;

}

else {

current = current->\_left;

}

}

return pre;

}

tr \* MainForm::prevElemTree22(tr \* cur\_root) {

if (cur\_root->\_left != NULL)

return maxElemTree(cur\_root->\_left);

tr \* parent = cur\_root->\_parent;

while (parent != NULL) {

if (parent->\_year < cur\_root->\_year) {

break;

}

parent = parent->\_parent;

}

return parent;

}

tr \* MainForm::maxElemTree(tr \* cur\_root) {

if (cur\_root == NULL) return root;

if (cur\_root->\_right == NULL) return cur\_root;

return maxElemTree(cur\_root->\_right);

}

Void MainForm::sort\_but\_Click(Object^ sender, EventArgs^ e) {

std::string str;

std::string bui;

str = msclr::interop::marshal\_as<std::string>(textBox\_street->Text);

bui = msclr::interop::marshal\_as<std::string>(textBox\_bui->Text);

int count=0;

if (isBound->Checked) {

bound = 1000;

}

else {

all = first;

while (all != NULL) {

if ((all->\_street == str) && (all->\_build == bui)) {

bound++;

}

all = all->\_next;

}

bound++;

}

all = first;

while (all != NULL) {

tr \* t;

if ((all->\_street == str) && (all->\_build == bui)) {

t = addElemToTree(&root, all, all->\_year\*bound + count);

count++;

}

all = all->\_next;

}

dataGridView1->Rows->Clear();

int i = 0;

tr \* cur\_root = maxElemTree(root);

while (cur\_root != NULL) {

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->Cells[0]->Value = Convert::ToString(cur\_root->item->\_no);

dataGridView1->Rows[i]->Cells[1]->Value = gcnew String(cur\_root->item->\_surname.c\_str());

dataGridView1->Rows[i]->Cells[2]->Value = gcnew String(cur\_root->item->\_name.c\_str());

dataGridView1->Rows[i]->Cells[3]->Value = gcnew String(cur\_root->item->\_pobatkovi.c\_str());

dataGridView1->Rows[i]->Cells[4]->Value = Convert::ToString(cur\_root->item->\_year);

dataGridView1->Rows[i]->Cells[5]->Value = gcnew String(cur\_root->item->\_area.c\_str());

dataGridView1->Rows[i]->Cells[6]->Value = gcnew String(cur\_root->item->\_street.c\_str());

dataGridView1->Rows[i]->Cells[7]->Value = gcnew String(cur\_root->item->\_build.c\_str());

dataGridView1->Rows[i]->Cells[8]->Value = gcnew String(cur\_root->item->\_flat.c\_str());

cur\_root = prevElemTree(cur\_root);

i++;

}}}

# Результати роботи програми

Результати роботи програми подамо у вигляді таблиці контрольних значень (табл. 1).

Таблиця 1. Таблиця контрольних значень

|  |  |  |
| --- | --- | --- |
| № тесту | Вхідні дані | Результати |
| 1. | Порожній список | Виконання команди INSERT та збереження нового файлу даних |
| 2. | Файл sr1.txt | Виконання команд INSERT та DELETE з різними параметрами |
| 3. | Файл sr1.txt | Сортування мешканців за віком з різними роками народження |
| 4. | Файл sr1.txt | Додавання мешканця з повторним роком народження і тестування команд INSERT, DELETE та сортування |

# Висновки

В ході виконання лабораторної роботи, було вивчено динамічні структури даних та основні операції над ними, задачі створення однозв’язного списку даних, заповнення двійкового дерева, пошуку найбільшого елемента в ньому, пошуку попереднього елементу, студенткою отримано навички створення алгоритму та написання програми, її тестування та створення протоколу лабораторної роботу. А виконана лабораторна робота буде розміщена на GitHub в якості одного з елементів портфоліо і допоможе студенці знайти класну роботу.