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

1. 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.

2. Алгоритм розв’язання завдання

1. Створення однозв’язного списку мешканців одного мікрорайону
    1.1. За допомогою файла-БД
    1.2. 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.

3. 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>(openFileDialog
1->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>(saveFileDialog
1->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++;
                      }}}
```

4. 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

5. 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.