НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Обчислювальної техніки та програмування»

**Звіт з практичної роботи №8**

Тема: «Списки»

Виконав:

Єрмаков М.В.

КІТ-120в

Дата: 27.04.21

Харків 2021

**Варіант 5**

**Хід роботи**

**Завдання:**

Для варіантів завдань 1 – 13 розробити програму, що створює список списків (нелінійний список). Передбачити такі функції:

− додавання елементів у список та підсписок (при додаванні елемента у головний список додається і відповідний підсписок);

− видалення елементів зі списку та підсписків (при видаленні елемента з головного списку видаляється і пов’язаний з ним підсписок);

− видача вмісту списків та підсписків у консоль;

Результат роботи програми

Індивідуальні завдання− видалення списків.

Завдання обрати у табл. 8.1 згідно зі своїм номером у журналі групи.

**Індивідуальне завдання:**

Список: «Маршрути поїздів». Підсписок: «Назви пунктів зупинок автобуса»

**Код:**

/\*\*

\* @author Ermakov M.

\* @date 27.04.2021

\* @brief EXERCISE №8

\*/

#include <iostream>

#include <stdlib.h>

#include <locale>

using namespace std;

typedef struct tree

{

string key;

struct tree\* left;

struct tree\* right;

struct tree\* parent;

} node;

node\* create(node\* root, string key);

node\* add(node\* root, string key);

void preorder(node\* root);

void Remove(node\* root, string value);

int main()

{

string \* autobuses = new string[sizeof(string)];

size\_t sizea;

cout << "Enter the quantity of rootes: ";

cin >> sizea;

for (size\_t i = 0; i < sizea; i++)

{

cout << "Name: ";

cin >> autobuses[i];

}

size\_t quanRootes;

cout << "Enter a quantity of races in each roote: ";

cin >> quanRootes;

string\*\* marchrutes = new string \*[sizea];

for (size\_t count = 0; count < sizea; count++)

marchrutes[count] = new string[quanRootes];

for (size\_t i = 0; i < sizea; i++)

{

cout << autobuses[i] << endl;

for (size\_t j = 0; j < quanRootes; j++)

{

cout << "City: ";

cin >> marchrutes[i][j];

}

cout << endl;

}

node\* root = new node[sizeof(node)];

node\*\* list = new node\*[sizeof(node\*)];

for (size\_t i = 0; i < sizea; i++)

{

list[i] = create(root, autobuses[i]);

}

cout << "-----------MAIN LIST OF ROOTES-----------" << endl;

for (size\_t i = 0; i < sizea; i++)

{

cout << "Roote No";

for (size\_t j = 0; j < quanRootes; j++)

{

if(list[i] != NULL)

add(list[i], marchrutes[i][j]);

}

preorder(list[i]);

cout << endl << "--------------------" << endl;

}

string enterRoote;

string deleteCity;

cout << "Enter the number of roote: ";

cin >> enterRoote;

int k;

for (size\_t i = 0; i < sizea; i++)

{

if (enterRoote == list[i]->key)

{

k = i;

}

}

cout << "Cities, which avaible for deleting: ";

preorder(list[k]);

cout << "Enter, which city you would like to delete here: ";

cin >> deleteCity;

node\* lst = list[k];

cout << endl << "----------------" << endl;

Remove(lst, deleteCity);

cout << endl << "----------------" << endl;

preorder(list[k]);

int delingRoote;

string delingRooteStr;

cout << endl << "Which one roote you would like to delete?: ";

cin >> delingRooteStr;

for (size\_t i = 0; i < sizea; i++)

{

if (delingRooteStr == list[i]->key)

{

delingRoote = i;

}

}

cout << endl << "----------------" << endl;

Remove(list[delingRoote], list[delingRoote]->key);

cout << endl << "----------------" << endl;

for (size\_t i = 0; i < (sizea); i++)

{

if (delingRoote != i) {

cout << "Root No";

preorder(list[i]);

cout << endl << "--------------------" << endl;

}

}

return 0;

}

node\* create(node\* root, string key)

{

node\* tmp = new node[sizeof(node)];

tmp->key = key;

tmp->parent = NULL;

tmp->left = tmp->right = NULL;

root = tmp;

return root;

}

node\* add(node\* root, string key)

{

node\* root2 = root, \* root3 = NULL;

node\* tmp = new node[sizeof(node)];

tmp->key = key;

while (root2 != NULL)

{

root3 = root2;

if (key < root2->key)

root2 = root2->left;

else

root2 = root2->right;

}

tmp->parent = root3;

tmp->left = NULL;

tmp->right = NULL;

if (key < root3->key) root3->left = tmp;

else root3->right = tmp;

return root;

}

void preorder(node\* root)

{

if (root == NULL)

return;

if (root != NULL)

cout << root->key << endl;

preorder(root->left);

preorder(root->right);

}

void Remove(node\* root, string value)

{

node\* pointer = root;

node\* parent = NULL;

while (pointer != NULL && pointer->key != value)

{

parent = pointer;

if (value < pointer->key)

pointer = pointer->left;

else

pointer = pointer->right;

}

if (pointer != NULL)

{

node\* removed = NULL;

if (pointer->left == NULL || pointer->right == NULL)

{

node\* child = NULL;

removed = pointer;

if (pointer->left != NULL)

child = pointer->left;

else if (pointer->right != NULL)

child = pointer->right;

if (parent == NULL)

root = child;

else

{

if (parent->left == pointer)

parent->left = child;

else

parent->right = child;

}

}

else

{

node\* mostLeft = pointer->right;

node\* mostLeftParent = pointer;

while (mostLeft->left != NULL)

{

mostLeftParent = mostLeft;

mostLeft = mostLeft->left;

}

removed = pointer;

pointer->key = mostLeft->key;

if (mostLeftParent->left == mostLeft)

mostLeftParent->left = NULL;

else

mostLeftParent->right = NULL;

}

cout << removed->key << " deleted" << endl;

delete[] removed;

}

}

**Приклад результату виводу:**

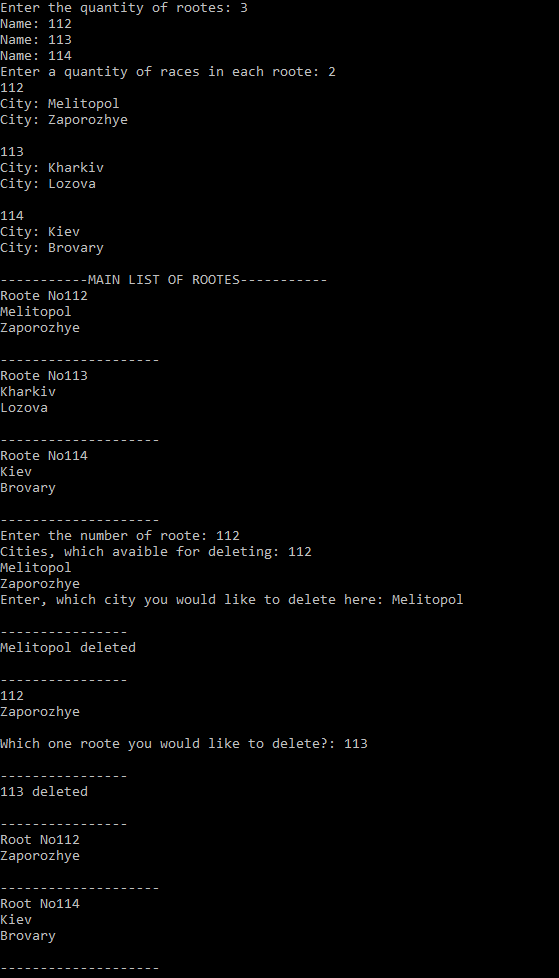
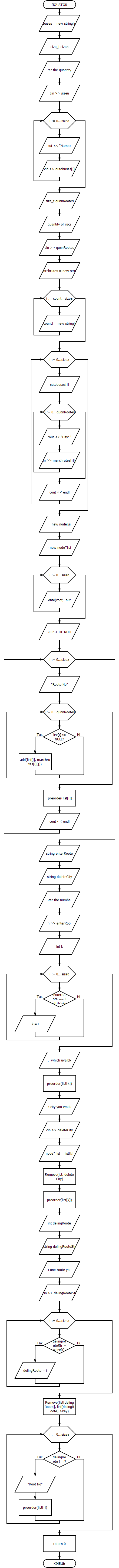


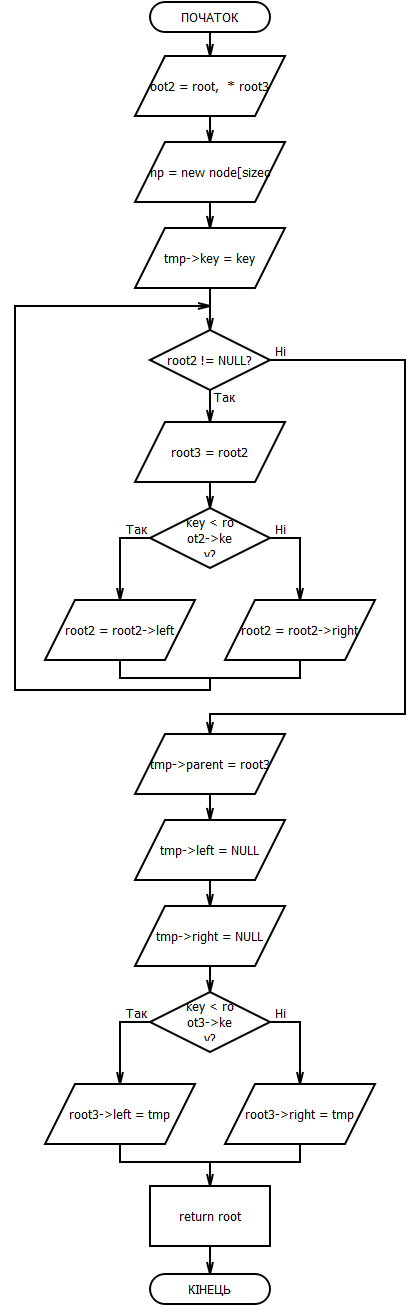
Рисунок 1 – Результат виводу програми

**Принцип дії програми:**

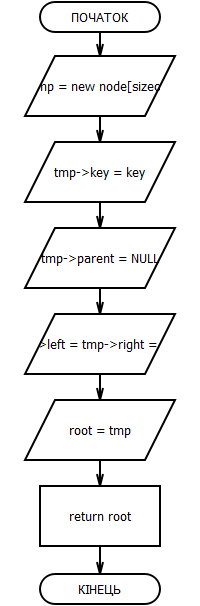
Перед початком програми створюємо структуру списку з наступним та попереднім показником й з цілим значенням. Починаємо виконання програми. Спочатку оголошуємо змінну типу int для визначення довжини нашого списку. Потім вводимо елементи масиву в список. Виводимо їх. Потім з кінця видаляємо кожний елемент. Виводимо кожне видалення в консоль.



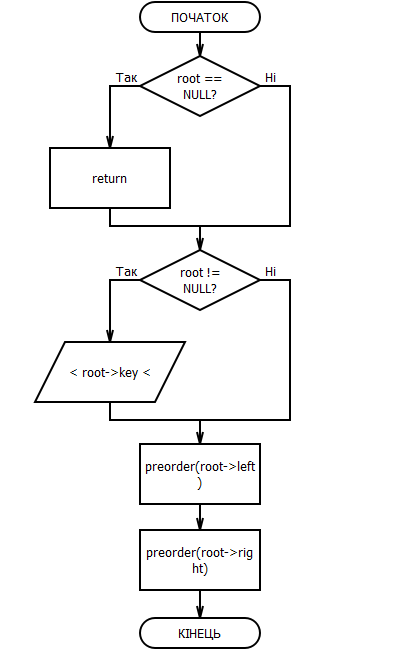
Блок-схема 1 – принцип роботи програми. Головна функція.



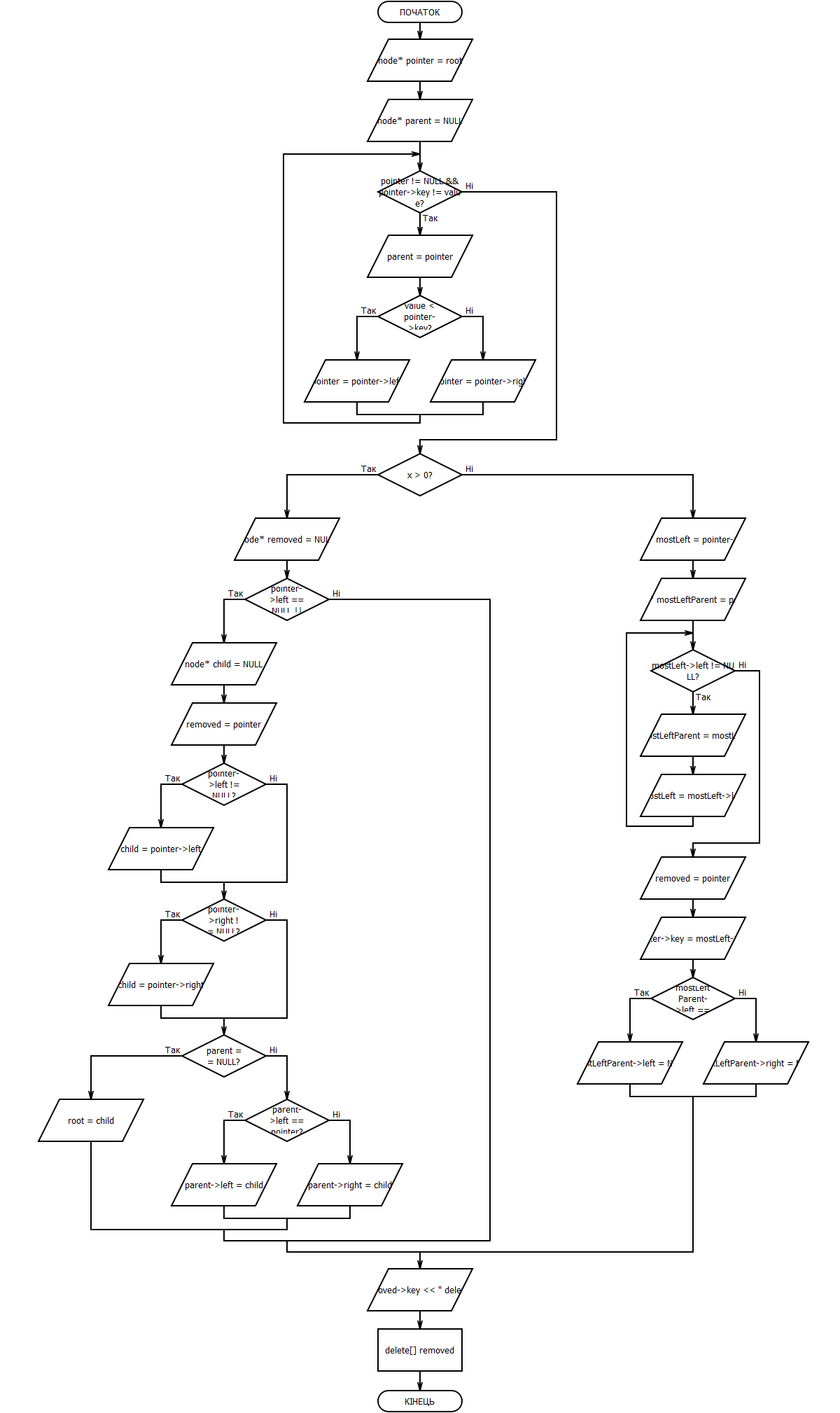
Блок-схема 2 – принцип роботи програми. Функція add()



Блок-схема 3 – принцип роботи програми. Функція create ().



Блок-схема 4 – принцип роботи програми. Функція preorder().



Блок-схема 5 – принцип роботи програми. Функція remove().

**Висновок:** я набув навичок та закріпив знання при виконанні операцій на мультисписках та нелінійних списках.